

AD-A065 909

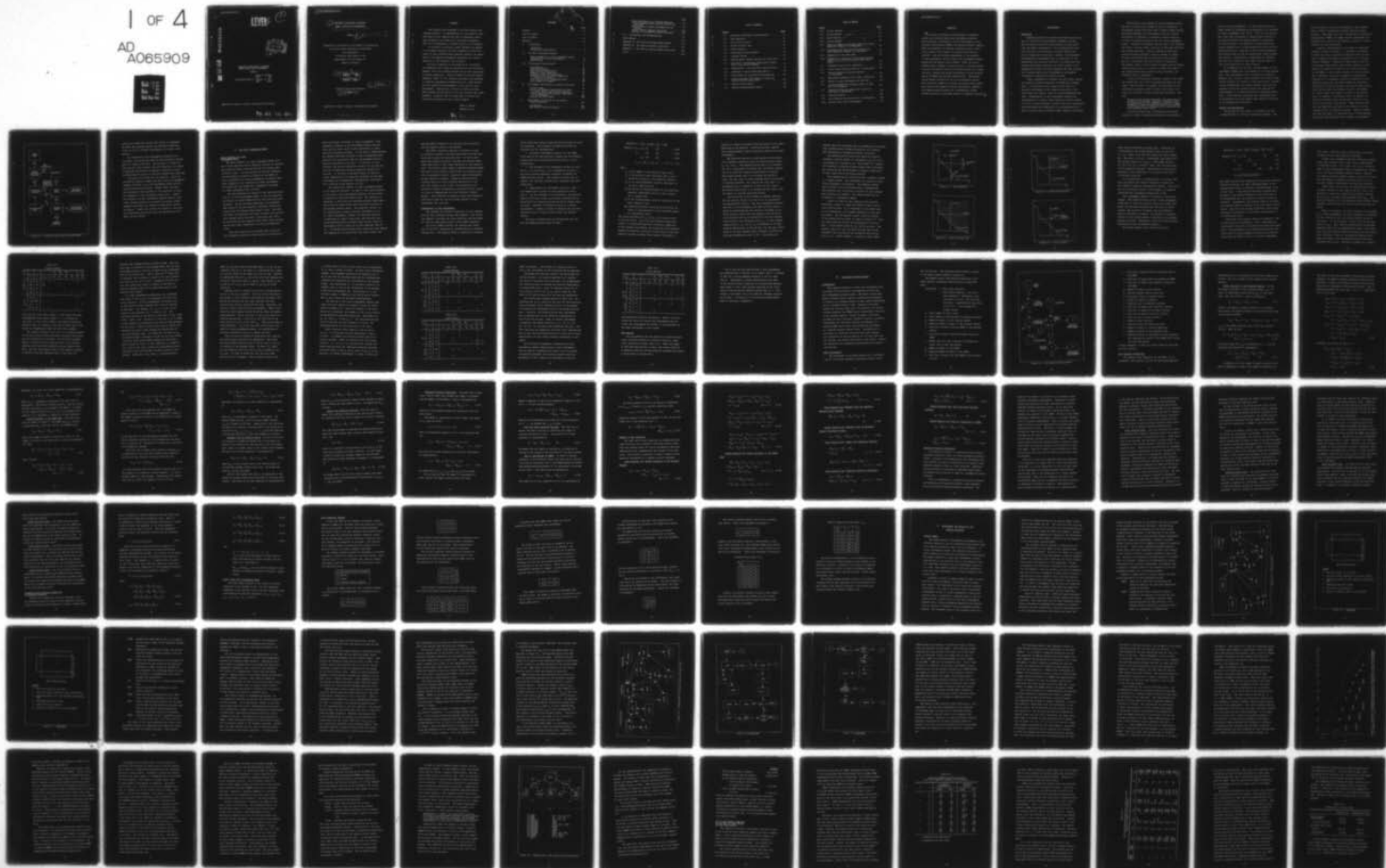
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 12/2
PERSONNEL CONTINGENCY PLANNING MODEL USING GOAL PROGRAMMING.(U)
DEC 78 J A MORENO, B W UTZ
AFIT/60R/SM/78D-10

UNCLASSIFIED

NL

1 OF 4

AD
A065909



LEVEL

(1)

AD A0 65909

DDC FILE COPY



PERSONNEL CONTINGENCY PLANNING
MODEL USING GOAL PROGRAMMING

THESIS

James A. Moreno
Capt USA

AFIT/GOR/SM/78D-10 Bradley W. Utz
Capt USAF

Approved for public release; distribution unlimited

79 03 12 081

6 PERSONNEL CONTINGENCY PLANNING
MODEL USING GOAL PROGRAMMING.

THESIS 9 Master's thesis

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

10 James A./Moreno
Captain USA
Bradley W./Utz
Captain USAF

12 302 p.

Graduate Operations Research

11 December 1978

Approved for public release; distribution unlimited

01 2 225

TUB

Preface

This research was performed to aid the military contingency planner. If implementation of any aspect of this work does in fact simplify the task of personnel contingency planning, our efforts will have been worthwhile, both for the Department of Defense and for us as students.

In an effort to provide a useful document as opposed to an academic exercise, we have attempted to write this report in plain English where possible; however, the mathematical and computer programming techniques and results essential to the discussion are included.

We wish to express our gratitude to Dr. Jon Knight for suggesting this study and providing his aid and encouragement, and to Dr. Kenneth Melendez for his able assistance in thesis preparation. Special thanks are due Capt Robert Shumacher, USAF, for his valuable assistance and knowledge in handling programming problems and to Dr. James Ignizio for his insight and encouragement in the realm of goal programming. Additionally, we wish to thank our wives, Cheryl Moreno and Shirley Utz, for their patience and understanding. A special note of thanks goes to Shirley Utz and Phyllis Reynolds for their typing support.

James A. Moreno

Bradley W. Utz

79 03 12 081

Contents

	Page
Preface	ii
List of Figures	v
List of Tables	vi
Abstract	vii
I. Introduction	1
Background	1
Purpose and Organization	3
II. The Goal Programming Method	7
Basic Elements of a Goal Programming Model	7
Fundamentals of Goal Programming	9
Applications	25
III. Contingency Planning Model	27
Introduction	27
Model Development	27
Goal Equation Formulation	30
Summary of Goal Equations	39
Objective Function Formulation	42
Preemptive Goal Structure Summarized in Objective Function	47
Final Linear Goal Programming Model	49
First Numerical Example	50
IV. Development and Testing of Simplex Algorithm	56
Initial Steps	56
Use of the Simplex Algorithm on the Case of Three Skills and Six Time periods	81
Further Algorithm Development: Employment of Core-Swapping	88
Running the Final Model	100
V. Development and Testing of the Pattern Search Algorithm	106
Introduction	106
The Pattern Search Procedure	106

	Page
Minor Modification of Problem Structure	111
General Comments About Pattern Search and Our Model	114
Modification of Search Procedure for Our Problem	121
Pattern Search Computer Algorithm	131
Testing of the Modified Search Procechure . . .	133
VI. Conclusions and Recommendations	152
Bibliography	157
Appendix A: The Recruiting Requirements Model	161
Appendix B: The Simplex Computer Algorithm	167
Appendix C: The Pattern Search Computer Algorithm . .	262
Vitae	290

List of Figures

<u>Figure</u>		<u>Page</u>
1-1	Personnel Contingency Planning Model	5
2-1	Goal Equations	14
2-2	First Priority Goal	14
2-3	Second Priority Goal	15
2-4	Final Solution	15
3-1	Contingency Planning Model	29
4-1	Original Design	59
4-2	Updated Model: Modular Design and Logic Flow . .	66
4-3	One Block of Technology Coefficients (Three Skill, Four Period Example)	73
4-4	Segmentation: Tree Structure and Directives . .	79
4-5	Breakdown of Split Technology Matrix	93
4-6	Final Model for Large-Scale Contingencies . . .	101
5-1	Conflict Between Achievement of Third and Fourth Priority Level Goals	128
5-2	Pattern Search Design	132
A-1	Recruiting Requirements Model	163

List of Tables

<u>Table</u>	<u>Page</u>
II-1 Initial Tableau	19
II-2 Second Tableau	23
II-3 Third Tableau	23
II-4 Fourth Tableau	25
IV-1 Resulting CONUS and Overseas Levels For First Numerical Example (in Thousands)	83
IV-2 Resulting Flows and Instructor Levels For First Numerical Example (in Thousands)	85
IV-3 Comparison of the Two Loads	87
IV-4 Comparison of Solutions: One Without Weights and One with Different Weights Among Skill Types	89
V-1 Parameters and Initial Conditions	134
V-2 Initial Base Training Pipeline (in Thousands)	135
V-3 Initial Technical Training Pipeline (in Thousands)	135
V-4 Additional Parameters and Initial Conditions	135
V-5 Extent of Goal Achievement for Test One	136
V-6 Resulting CONUS and Overseas Levels of Test One (in Thousands)	137
V-7 Resulting Flows and Instructor Levels for Test One (in Thousands)	139
V-8 Testing Scenario	146
V-9 Five Scenario Testing Results (in Thousands)	148
V-10 Percent Force Level Achievement	150

Abstract

↓ A military contingency planning model is designed, tested, and evaluated using goal programming analysis. This prototype is designed to aid the achievement of Continental United States (CONUS) and overseas theater requirements for personnel of different categories over an arbitrary length of time subject to resource and time constraints. The two alternate approaches of the research in goal programming are the linear goal programming simplex method and the pattern search method. Tests are run on somewhat small examples--a three-job skill, six-time period case and six priority levels of projected goals and a three-job skill, twenty-time period contingency of seven priority levels of CONUS and overseas theater goals. The pattern search method (tailored to fit to the particular policy goal structure) appears to provide the analyst with reasonable results and computer resource conservation, whereas the simplex method provides for a mathematical global optimum solution at increased expense in computer resources. ↙

I. Introduction

Background

Peacetime planning for wartime requirements has been a part of the military establishment throughout history, and this thesis focuses on that planning for both peacetime and combat contingencies. The considerations which are involved in this planning are indeed numerous and tend to complicate the process. However, the development of mathematical programming from its beginnings in the Second World War has been effective in streamlining these planning procedures; moreover, the recent growth in the computational power of digital computers has made feasible the increased application of mathematical programming techniques to improve the planning process.

Linear programming is probably the most widely applied and carefully studied technique of mathematical programming. Its practical nature stems directly from G. B. Dantzig's development of the simplex algorithm in 1947 which solves the general class of linear programming problems. Fundamentally, linear programming requires the optimization of a single linear objective function subject to a set of linear resource, process, and policy constraints. The idea behind this formulation is the determination of how to allocate scarce resources among competing activities.

Unfortunately, the problem for the contingency planner may not be realistically reduced to such a formulation. A contingency planner often must make a tradeoff among several conflicting measures of effectiveness, and, as a result, he normally is reluctant to determine a single criterion to use as an objective function for the linear programming format. Instead, he determines several goals to be achieved and tries to satisfy them in the face of both goal conflicts and limited resources. Ideally, resolution comes when the decision maker develops a priority scheme or ranking of these goals based upon his particular value system. Once he is satisfied with this ordering of objectives, the planner tries to fulfill goals of higher priority before lower-priority goals; as a result, he finds that some of the lower-priority goals cannot be fully achieved, and tradeoffs must be introduced.

In real world planning situations, however, this trade-off process may not be fully developed and explicitly followed. The Air Force, for example, has developed its own professional planning staffs to assure that viable solutions are available to answer operational questions. However,

because of the limited functional and organizational horizons of each staff, potential tradeoffs cannot be recognized--and the information required to make tradeoffs is often filtered out before it is passed up (the hierarchy to coordinating and consolidating staff agencies) [Ref 18:27].

A course of action that a contingency planner may consider in order to generate solutions to his problem is

the use of goal programming. In 1961 A. Charnes and W. W. Cooper formally introduced goal programming as an extension to linear programming (Ref 19:16). Prior to this development, more than one measure of effectiveness involved in the objective forced the decision maker to reformulate his entire problem. All measures of effectiveness were usually combined into a single measure--utility; with this single measure and a familiarity with linear programming, the decision maker solved the problem with linear programming methods. Unfortunately, transforming the measures into utility is difficult and clumsy; reality offers no such single objective for every occasion.

To accommodate this reality, the development of goal programming stemmed from the foundations of linear programming. Goal programming tries to consider all objective measures that a planner thinks important and allows a simultaneous solution to a system of conflicting goals. Without a prior reformulation of a single measure, this solution procedure treats the tradeoff process and requires the establishment of an ordinal hierarchy of importance among goals. (The hierarchy stipulates that lower-order goals are considered only after higher-order goals are fulfilled to the maximum extent possible.)

Purpose and Organization

The purpose of this thesis is to develop and test a proposed model for military contingency planning. The

plan of attack is to use the techniques of goal programming in order to develop and evaluate various planning policies. This model will aid in the meeting of Continental United States (CONUS) and combat theater requirements for personnel for a specific time duration, subject to various resource and time constraints.

A general representation of the model for personnel planning is shown in Figure 1-1. The model is designed to chart the flows of personnel resources within the military to support contingency requirements for an overseas theater engagement. Supplies of human capital come from the national labor market, the military reserves, and the quantity of personnel already in the system (stationed in the CONUS, in the overseas theater, in basic training, and in technical training as students or instructors). Planning for an engagement involves consideration of the demand for particular job skills needed for operations, the duration of the contingency, and several other factors under and not under the planner's control. An example of a factor under the planner's control is the desired ratio of students to instructors, while an example of a factor not normally under the planner's control is the size of the national labor market. Human resources flow into the CONUS force and then to the theater or to the instructor pool as conditions warrant. Return of personnel from the theater or instructor pools is also possible. Attritions from training and the theater are provided to

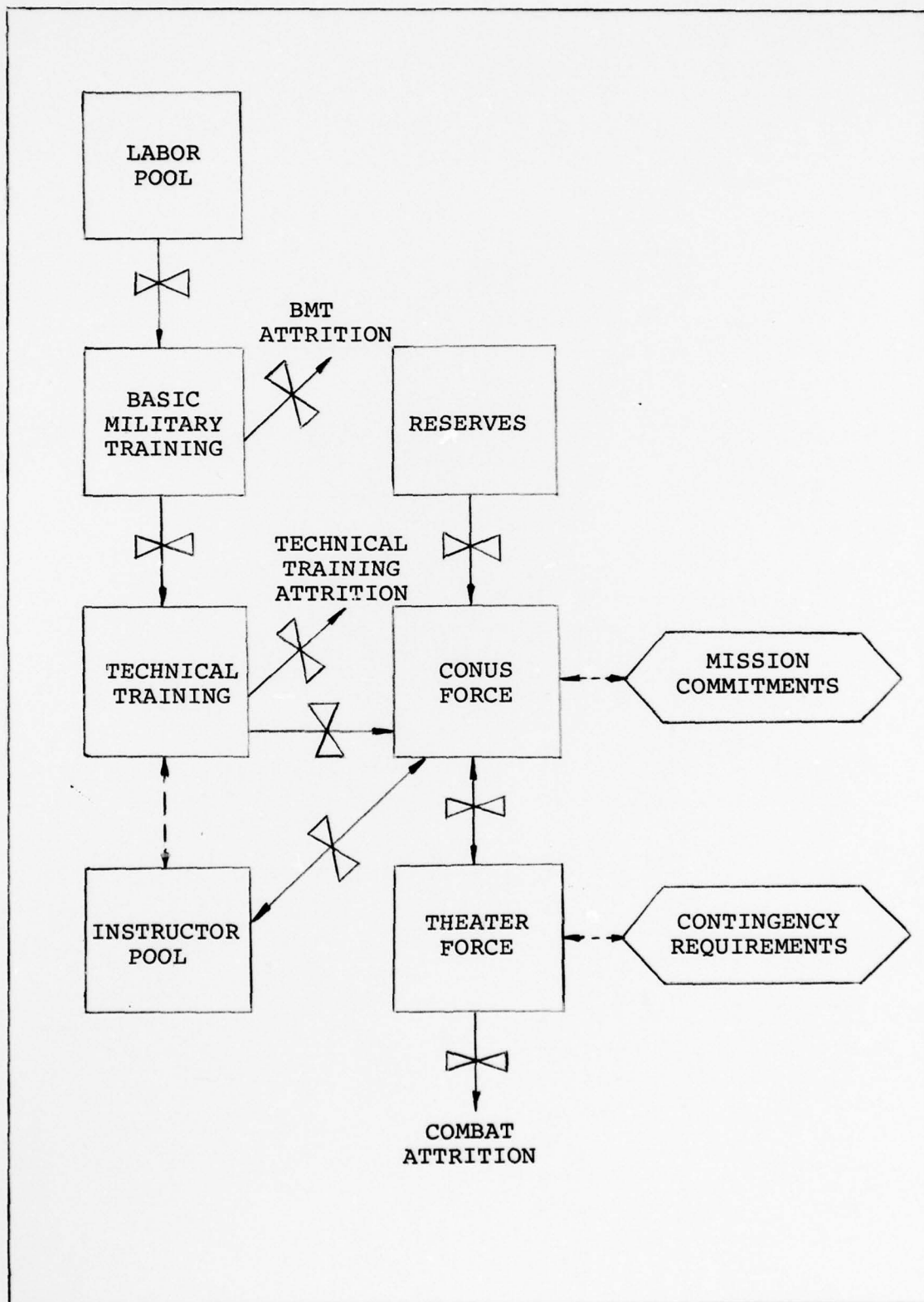


Figure 1-1. Personnel Contingency Planning Model

reflect the losses that would occur during an engagement. The model may consider several job categories simultaneously, as well as consider requirements for several time periods.

By incorporating goal programming procedures into the model, the planner may identify and distinguish between goals of varying priorities. As the model is tested for more and more time intervals and job categories, the number of equations and decision variables is expected to grow, necessitating the use of an efficient computerized algorithm.

The thesis is divided into six chapters. After the introductory chapter, Chapter Two deals with the fundamental elements involved in the goal programming method and discusses previous applications. Chapter Three describes the Contingency Planning Model of this research in detail with emphasis on goal programming application. Chapter Four presents the goal programming simplex computer algorithm, whereas Chapter Five presents the pattern search computer algorithm; both chapters include model testing procedures. Chapter Six reviews the results of the tests and establishes conclusions about the usefulness of the model and the method.

II. The Goal Programming Method

Basic Elements of a Goal Programming Model

The basic elements of a goal programming model are a set of goal equations and an objective function (or achievement function) which evaluates how well the overall goal structure is met. As the structure of the goal equations and the objective function is similar to the format of their counterparts in linear programming, we discuss these basic elements in the vocabulary of linear programming and assume that the reader has a fundamental knowledge of linear programming (Ref 13:15-97).

The goal equations are similar to the constraint equations of a linear programming model in that the equations are written as linear combinations of the various decision variables. Unlike constraint equations, the righthand-side values of the goal equations represent various goal levels which the decision maker wants to meet. The decision maker can determine the extent to which a set of decision variable values satisfies a particular goal level by evaluating the goal equation with these values and comparing this linear combination with the righthand-side value.

Every goal equation also includes both a positive and a negative deviation variable which act similarly to

slack and surplus variables in linear programming. Each represents a measure of the discrepancy between the goal level and the actual level of goal achievement by these particular decision variables. If the goal is underachieved, the negative deviation variable of that corresponding goal equation will be positive (and the corresponding positive deviation variable will be zero). If the goal is overachieved, the positive deviation variable will be positive (and the corresponding negative deviation variable will be zero). If the goal is achieved exactly so that the linear combination of the decision variables for the equation equals the corresponding righthand-side, both deviation variables of the equation are zero.

The second basic element of a goal programming model is the objective function. To adapt a real world situation to the class of goal programming problems, we must create a hierarchy of importance among goals. The hierarchy is necessary in order that the algorithm consider lower-order goals once the higher-order goals have been satisfied to the maximum possible extent. The objective function is not constructed in terms of decision variables as it is in linear programming; instead, the objective function is described in terms of deviations (the deviation variables mentioned earlier) between goals and the levels of achievement within a given set of goal equations (Ref 19: 22). To deviate more and more from a particular goal implies the tendency not to achieve that goal fully; hence, the

decision maker's purpose is to minimize these deviations from the specific goals of the problem.

The model's objective function is written in terms of deviation variables, and those deviations from the highest priority goals are minimized first. If two or more goals have identical priority rankings (the decision maker finds these goals to bear equal importance to the organization), the decision maker imposes a modified ordering scheme. Each goal at this priority level within the objective function is weighted by having a numerical quantity multiplied by each deviation variable. Weighting these deviation variables does not create differing priorities for one goal over another at this level; the algorithm is constructed to handle the deviations from goals within a priority class as commensurable but with different tradeoff weights. This technique of weighting along with preemptive priorities allows for very fine-tuned analysis of goal achievement (Ref 19:26-30).

Fundamentals of Goal Programming

The concepts and solution techniques for goal programming are best explained through some examples. The following examples illustrate the setup conditions and the graphical solution approaches as well as the simplex approach.

Our first example involves the training and allocation of two skill categories of enlisted men by a technical training unit. The decision maker is required to establish

goals concerning training loads and the utilization of training resources. The following information and data are available for his consideration:

1. Personnel requirements forecast for the CONUS force and for the time period of interest are six thousand persons and eight thousand persons with skills 1 and 2, respectively.

2. Ten thousand of the individuals (either all with skill 1, all with skill 2, or a combination of both) is the total training capability available to support the CONUS personnel quotas. This capability refers only to regular, not overtime, working hours (necessary day shift conditions).

3. Requirements for individuals with skill type 1 are deemed twice as urgent as those demands for skill type 2. For the alignment of his objectives, the decision maker believes that he must give primary consideration to maintaining full capacity utilization during regular working hours. Secondly, he desires to meet all training requirements. Finally, he wants to minimize any overtime operations (use of crews on shifts other than daytime shifts).

The above considerations are incorporated into the goal programming problem which follows:

$$\begin{aligned}
& \text{Minimize } Z = P_1 d_1^- + P_2 (2d_2^- + d_3^-) + P_3 d_1^+ \\
& \text{Subject to } x_1 + x_2 + d_1^- - d_1^+ = 10000 \\
& \quad \quad \quad x_1 + d_2^- - d_2^+ = 6000 \\
& \quad \quad \quad x_2 + d_3^- - d_3^+ = 8000 \\
& \quad \quad \quad x_i \geq 0, d_j^+ \geq 0, d_j^- \geq 0 \quad i = 1, 2; j = 1, 2, 3
\end{aligned}$$

where

x_i is the number of individuals of skill type i

P_i is the priority level associated with a set of deviation variables in the objective function

d_1^- is the underachievement (negative deviation) of the total capability goal

d_2^- and d_3^- are the underachievement of the predicted personnel requirements goals for skill types 1 and 2, respectively

d_1^+ is the overachievement (positive deviation) of the total capability goal

The remaining deviation variables d_2^+ and d_3^+ are the levels of overachievement of the predicted personnel requirements goals.

The choice variables are x_1 and x_2 , the training levels of the two skills. The first goal equation states that there is a regular-working-hours utilization capability of ten thousand individuals; the second and third equations express the personnel forecasts for both skills; each goal equation includes variables which measure the amount of

positive or negative deviation from the values on the right-hand-side of the equation. (Positive deviation implies goal overachievement; negative deviation implies goal underachievement.)

The objective function is constructed from the deviation variables relating to goals in three priority levels-- P_1 , P_2 , and P_3 . The highest priority goal is to minimize d_1^- ; this is equivalent to stating that utilization capability of at least full regular working hours is desired. The second priority goal is to minimize the quantity $(2d_2^- + d_3^-)$ and, thus, to effect maximum levels of personnel in training; moreover, the training for skill type 1 is considered twice as important as that for skill type 2. The third priority goal is to minimize d_1^+ and, hence, keep the amount of overtime operations minimal.

The priorities of the problem are defined ordinally, and the priority factors P_j 's are not to be interpreted as multiplicative factors. The goal programming objective function is to be minimized, and these priority factors which are associated with the deviation variables at different priority levels have the property that the deviation variables with P_j are minimized to the best possible extent prior to the variables with P_{j+1} . In this example problem with P_1 , P_2 , and P_3 , d_1^- is minimized to the best possible extent prior to the sum $2d_2^- + d_3^-$; the sum is minimized to the best possible extent (without violating the previous minimization) prior to d_1^+ . This definition

assures that each succeeding goal is optimized to the extent that preceding goals already attained are not violated. This constitutes a "preemptive priority structure."

The second priority term of the objective function includes a relative weight of 2 assigned to d_2^- and a relative weight of 1 assigned to d_3^- . This weight of 2 enables the algorithm to minimize d_2^- more quickly than to minimize d_3^- , reflecting the decision maker's ranking of importance of skill types within the second priority.

This example may be solved by either a graphical method or a simplex method. Both techniques are similar to their linear programming counterparts. The graphical method (Ref 19:64-92) will be presented since it demonstrates how deviations and the priority structure of goal programming functions are handled. The simplex method will be discussed in a subsequent example.

Figure 2-1 shows the goal equations with possible deviations; Figure 2-2 shows the feasible region for the first priority goal. d_1^- is minimized in the shaded area, as its minimization is of highest importance to the decision maker. The new feasible region, with the addition of the second priority goals, is shown in Figure 2-3. This new region does not violate the attainment of the first goal. In the shaded region d_1^- , d_2^- , and d_3^- are zero. The feasible region for only the third priority goal is the triangular area enclosed by the coordinate axes and the line $x_1 + x_2 = 10000$; however, a solution in this region

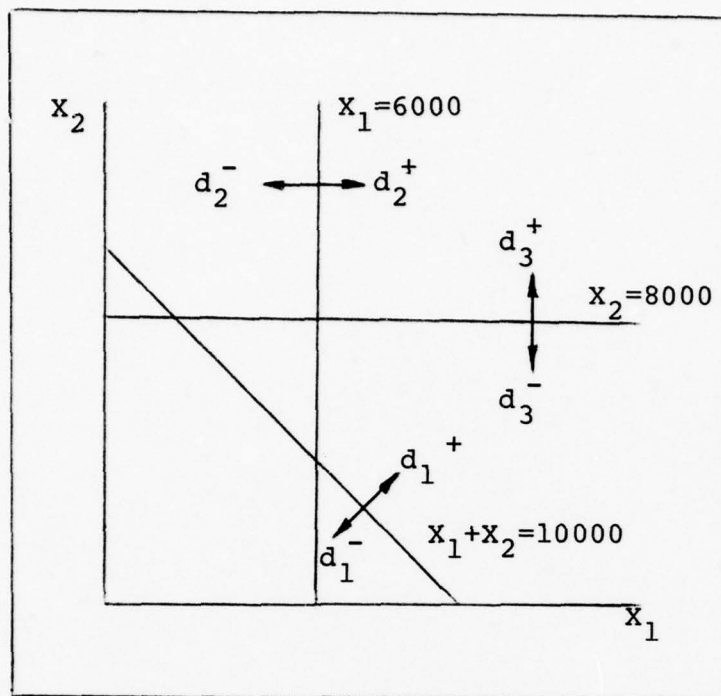


Figure 2-1. Goal Equations

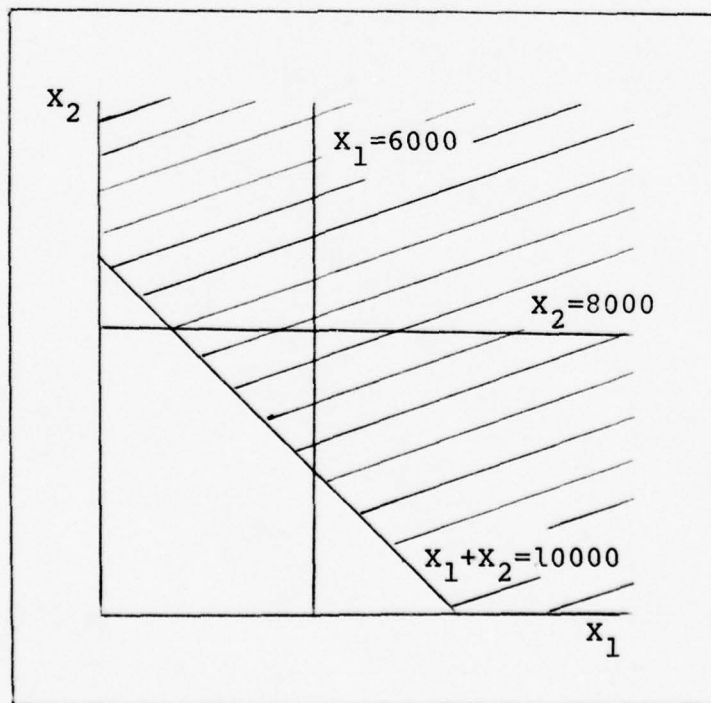


Figure 2-2. First Priority Goal

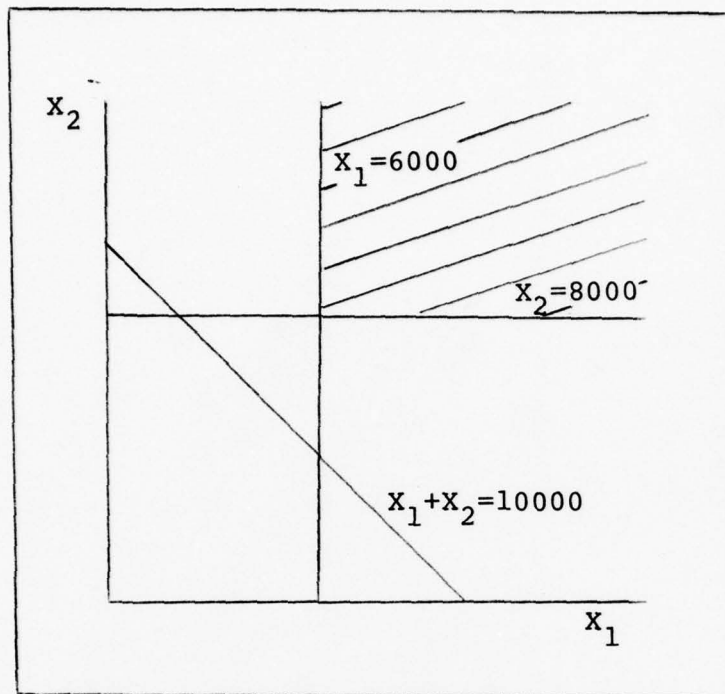


Figure 2-3. Second Priority Goal

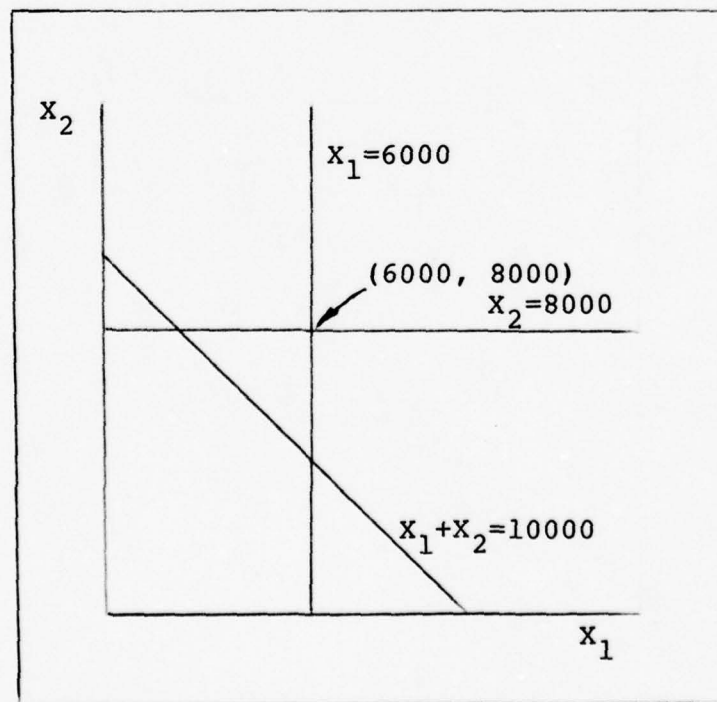


Figure 2-4. Final Solution

would violate previously attained goals. Therefore, d_1^+ is minimized at the closest possible point to the line $x_1 + x_2 = 10000$, which does not violate the goals already met. The point is $(x_1, x_2) = (6000, 8000)$ (see Figure 2-4). The optimal solution is $(x_1, x_2) = (6000, 8000)$ with $d_1^- = d_2^- = d_3^- = d_2^+ = d_3^+ = 0$ and $d_1^+ = 4000$. Only the third goal is not completely attained. Modification of the priority structure or the introduction of new goals could result in different solutions.

It is significant to recognize that, in no case, were both the positive and negative deviation variables for any constraint equation simultaneously nonzero. This situation is not possible and will never happen in a properly formulated goal programming problem (Ref 19:24).

Goal programming simplex is discussed in the second example. This method differs from its linear programming counterpart in that the objective function is always minimized, the deviation variables take the place of slack, surplus, and artificial variables, and the preemptive priority structure is ensured. The sequence of required actions in this simplex method is explained as the example problem is solved. (References 15 and 19 provide detailed discussions of goal programming simplex.)

The second example comes from Ref 19:97-104.

$$\text{Minimize } Z = P_1 d_1^- + P_2 d_4^+ + P_3 (5d_2^- + 3d_3^-) + P_4 d_1^+$$

$$\text{Subject to } x_1 + x_2 + d_1^- - d_1^+ = 80$$

$$x_1 + d_2^- = 70$$

$$x_2 + d_3^- = 45$$

$$x_1 + x_2 + d_4^- - d_4^+ = 90$$

$$x_1, x_2, d_1^-, d_2^-, d_3^-, d_4^-, d_1^+, d_4^+ \geq 0$$

The second and third goal equations do not contain positive deviation variables, for while underachievement of the righthand-side values is possible, overachievement is not in this case. If the positive deviation variables were included in the problem formulation, they would have to be incorporated into the objective function as priority one goals. Consequently, the original terms in the objective function would be assigned to the next lower priority levels. Then the first priority would be to drive these new positive deviations to zero before concentrating on the lower-order goals.

Before the first simplex tableau is presented, a few relevant points are considered. First, the purpose of the objective function is to minimize the total unattained goals. This minimizing of the associated deviation variables occurs by using preemptive priorities and weights, which take the place of the c_j 's of linear programming

(the linear constants used as coefficients of decision variables in the linear objective function).

Second, since preemptive priority factors are ordinal rather than cardinal values, they are not commensurable at different levels. As a result, the " $z_j - c_j$ " criterion used in linear programming cannot be expressed by a single row in the goal programming simplex tableau. Instead, the simplex criterion becomes a matrix of size m by n with m as the number of preemptive priority levels and n as the total number of both choice and deviation variables.

Third, since the criterion for determining the entering basic variable is now a matrix and not a row, a new procedure is required to identify the entering variable. As $P_j \gg P_{j+1}$, P_j always takes precedence over P_{j+1} . The selection procedure for the optimum entering variable column, therefore, must consider the level of priorities (Ref 19: 98-99).

Table II-1 presents the initial tableau of the example. As in linear programming, the basic assumption in establishing this tableau is that the initial solution occurs at the origin. At this position, values of all choice variables are zero. This condition characterizes the maximum underachievement of the righthand-side values, and, therefore, only the negative deviation variables are the initial basic variables. Since all the negative deviation variables are nonzero, the corresponding positive deviation variables must be zero. The basic variables are listed in

Table II-1
Initial Tableau

c_j			P_1 $5P_3$ $3P_3$ P_4 P_2							
	V	C	x_1	x_2	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_4^+
P_1	d_1^-	80	1	1	1				-1	
$5P_3$	d_2^-	70	(1)			1				
$3P_3$	d_3^-	45		1			1			
	d_4^-	90	1	1				1		-1
$(z_j - c_j)$	P_4	0							-1	
	P_3	485	5	3						
	P_2	0								-1
	P_1	80	1	1					-1	

the first four rows under column V in the table with their corresponding values under column C. It is always the rule that, in the initial tableau, the negative deviation variables (d_i^-) will appear in the solution base (Ref 19:99). Zeroes are suppressed to make the table more readable.

The simplex criterion ($z_j - c_j$) is a 4 x 8 matrix because of the four priority levels and eight variables--two choice plus six deviation--in the model. The solution procedure begins to achieve the most important goal to the fullest possible extent and then considers the next highest order goal. The unit contribution rate of each nonbasic variable in achieving the most important goal is the basis for

selecting the optimum entering variable column. Once the first goal is attained to the maximum extent then the optimum column selection criterion is based on the achievement rate for the second goal. Hence, when the j^{th} order goal is attained, this criterion is based on the unit contribution rate for the $(j+1)^{\text{th}}$ goal. Preemptive priority factors are listed from lowest to highest in the table so that the optimum column can be identified easily at the bottom of the tableau.

In linear programming minimization, the righthand-side value of the $(z_j - c_j)$ row represents the total cost of the solution. In goal programming, the values of P_j in the constant column represent the *unattained* portion of each goal. For example, $x_1 = x_2 = 0$ in the initial solution, and therefore, $d_2^- = 70$ and $d_3^- = 45$. The unattained portion of the priority three goal is $5d_2^- + 3d_3^- = 5(70) + 3(45) = 485$. Similarly, if one unit of x_1 is added to the solution, d_2^- decreases by one unit and $5d_2^-$ decreases by five units. Such an action would result in moving the solution five units closer towards the satisfaction of the third priority goal. By adding one unit more of x_2 , the problem shows a decrease in d_3^- by one unit and effects a shift of three units towards meeting the third goal. These entries of 5 and 3 in the respective x_i columns of the P_3 row of the $(z_j - c_j)$ matrix are calculated directly from rows 2 and 3 of the basic variable section of the tableau. (From row 2, $5P_3$ under c_j is multiplied by 1

under x_1 with the result positioned under x_1 in the P_3 row; likewise, from row 3, $3P_3$ under c_j is multiplied by 1 under x_2 , and the result is put under x_2 in the P_3 row). Nothing is subtracted from the c_j row at the top of the tableau, for entries of zero exist above x_1 and x_2 . The entries in the P_1 row of the $(z_j - c_j)$ matrix under x_1 and x_2 are found similarly.

Another point to consider is the set of column entries for d_1^- in the $(z_j - c_j)$ matrix. With the initial solution at the origin, a unit increase in d_1^+ yields an increase in d_1^+ , moving the solution one unit away from both the first priority goal and the fourth priority goal. Therefore, the entries in that column for d_1^+ are reasonable. Again, this event will never happen because d_1^+ and d_1^- cannot be nonzero simultaneously. The P_1 row entry in the $(z_j - c_j)$ matrix is found by multiplying -1 under d_1^+ in the first row of the basis matrix by the P_1 value under c_j . (The P_4 row entry is found similarly.) As is the case with initial basic variables of linear programming simplex, all coefficients of the basic variables in the $z_j - c_j$ matrix are zero.

Once the initial tableau is constructed, the entering and leaving basic variables are determined. The column with the largest positive value at the P_1 level in $(z_j - c_j)$ is selected as the optimum column. In Table II-1 there are two identical entries in the x_1 and x_2 columns at the P_1 level. In order to break this tie, the next order priority level is checked. Since a greater value in the

x_1 column exists at level P_3 than that value corresponding to x_2 , the x_1 column is chosen. As with linear programming simplex, the programmer determines the departing row to be the one with the minimum value when the righthand-side constants are divided by the coefficients of the optimum column. The coefficient of 1 is circled in Table II-1 to indicate that it lies at the intersection of the optimum column and the departing row. The entrance of x_1 into the basis causes a reduction in d_1^- and d_2^- and an improvement in goal levels one and three simultaneously.

With the use of the linear programming simplex algorithm, the first tableau is revised to yield the second tableau in Table II-2. After the elements in the basis matrix are calculated, the elements in the $(z_j - c_j)$ matrix are determined as previously explained. From Table II-2 we see that $x_1 = 70$ and $x_2 = 0$. The unattained portion of the first goal has been reduced to 10 units, and unattained portion of the third goal to 135 units.

Since a positive entry remains in the P_1 row of $(z_j - c_j)$, it is possible for more improvement in goal 1 to occur. x_2 becomes the entering variable and d_1^- the leaving variable. Table II-3 presents the third stage solution. $x_1 = 70$, $x_2 = 10$, and the first, second, and fourth priority goals are completely achieved; moreover, since each entry in the P_1 and P_2 rows of $(z_j - c_j)$ is non-positive, no further improvements in either of these two

Table II-2
Second Tableau

c_j			P_1 $5P_3$ $3P_3$ P_4 P_2							
	V	C	x_1	x_2	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_4^+
P_1	d_1^-	10		(1)	1	-1			-1	
	x_1	70	1			1				
$3P_3$	d_3^-	45		1			1			
	d_4^-	20		1		-1		1		-1
$(z_j - c_j)$	P_4	0							-1	
	P_3	135		3		-5				
	P_2	0								-1
	P_1	10		1		-1			-1	

Table II-3
Third Tableau

c_j			P_1 $5P_3$ $3P_3$ P_4 P_2							
	V	C	x_1	x_2	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_4^+
$3P_3$	x_2	10		1	1	-1			-1	
	x_1	70	1			1				
	d_3^-	35			-1	1	1		1	
	d_4^-	10			-1			1	(1)	-1
$(z_j - c_j)$	P_4	0							-1	
	P_3	105			-3	-2			3	
	P_2	0								-1
	P_1	0			-1					

goals is possible. Since there is a positive entry in the P_3 row, achievement of the third goal may be improved.

d_1^+ becomes the entering variable, and d_4^- becomes the leaving variable. Our present objective is to improve the level of satisfying this third goal, but accomplishing this objective must not degrade the existing achievements of higher-order goals. Since no negative entries lie in the P_1 and P_2 rows of the $(z_j - c_j)$ matrix under d_1^+ , the entry of d_1^+ into the basis will not pose any degradation.

The fourth-stage tableau appears in Table II-4. By examining the $(z_j - c_j)$ matrix we see that the third priority goal achievement can be improved but only at the expense of worsening the achievement level of the second priority goal. Similarly, the fourth priority goal achievement can be improved but only by effecting a degradation in the third priority goal. Thus, no further achievement is possible; the optimal solution is $x_1 = 70$, $x_2 = 20$, $d_3^- = 25$, $d_1^+ = 10$, and other goal deviations are zero. The first two goals are completely attained, while underachievement of the third and fourth goals by 25 units and 10 units, respectively, are the closest possible attainments we can expect.

As with linear programming, procedures have been developed to handle complications in goal programming, such as negative righthand-side values, ties in entering or departing variables and alternate optimal solutions. References 19 and 15 detail such procedures as well as

Table II-4
Fourth Tableau

c_j			P_1 $5P_3$ $3P_3$ P_4 P_2							
	V	C	x_1	x_2	d_1^-	d_2^-	d_3^-	d_4^-	d_1^+	d_4^+
$3P_3$ P_4	x_2	20		1		-1		1		-1
	x_1	70	1			1				
	d_3^-	25				1	1	-1		1
	d_1^+	10			-1			1	1	-1
$(z.-c.)$	P_4	10			-1			1		-1
	P_3	75				-2		-3		3
	P_2	0								-1
	P_1	0			-1					

post-optimality analysis and duality. Ignizio (15) discusses the topics of integer goal programming and non-linear goal programming but neither is incorporated into the model development in this thesis.

Applications

Goal programming has been applied to several decision areas including problems of resource allocation, scheduling, and policy analysis (Ref 15:2). While the number of published applications is small, the recent evolution of computer codes for solving practical problems has created a rapid growth in applications.

One of the earliest applications of goal programming was formulated by A. Charnes, W. W. Cooper, and R. J. Niehaus in 1967 for civilian manpower planning in the U.S. Navy (Ref 4). Subsequently, several modifications were made to the original model, resulting in the Recruiting Requirements Model in 1971, which Niehaus depicted as the "work-horse of the operational modeling system [Ref 21]." Variations of this model are in use today for manpower planning by the Navy. A discussion of the Recruiting Requirements Model is provided in Appendix A.

III. Contingency Planning Model

Introduction

This chapter describes in detail the Contingency Planning Model of this research with emphasis on use of the goal programming solution method. The model is developed under a scenario which requires a buildup and sustainment of forces in an overseas theater to meet combat requirements over an extended length of time. Sources of personnel include the civilian labor pool, the reserves, the initial training pipeline, the CONUS force, and personnel initially in the overseas theater. Within the model, provisions are made for basic training, technical training, instructor requirements for technical training, maintenance of a required CONUS force level, and buildup and sustainment of a desired overseas theater force. Consideration is also given to initiation of the military draft system, the limitations on the availability of civilian labor and the reserves, and various attritions in the system. Several job categories are considered simultaneously over several time periods.

Model Development

The development of the model begins with a consideration of its various components describing levels, flows,

and restrictions. The discussion which follows is keyed to the model network diagram in Figure 3-1.

The symbols used in the network diagram and in the model equation formulations which follow are described below.

Subscripts: i - Skill type indicator

t - Time period indicator. For stock type variables, t designates a specific point in time. For flow type variables, t indicates the time interval $(t-1, t)$. t will assume only integer values.

n - Total number of skill levels

h - Total number of time periods (or planning horizon)

T - Number of people in the overseas theater

\bar{T} - Required number of people in the overseas theater

x - Flow rate of people from the CONUS to the overseas theater

c - Flow rate of people from the overseas theater to CONUS

α - Combat attrition rate (fraction of on-hand personnel lost in one time period)

K - Number of people in the CONUS

\bar{K} - Required number of people in the CONUS

v - Flow rate of people from the CONUS to the instructor pool

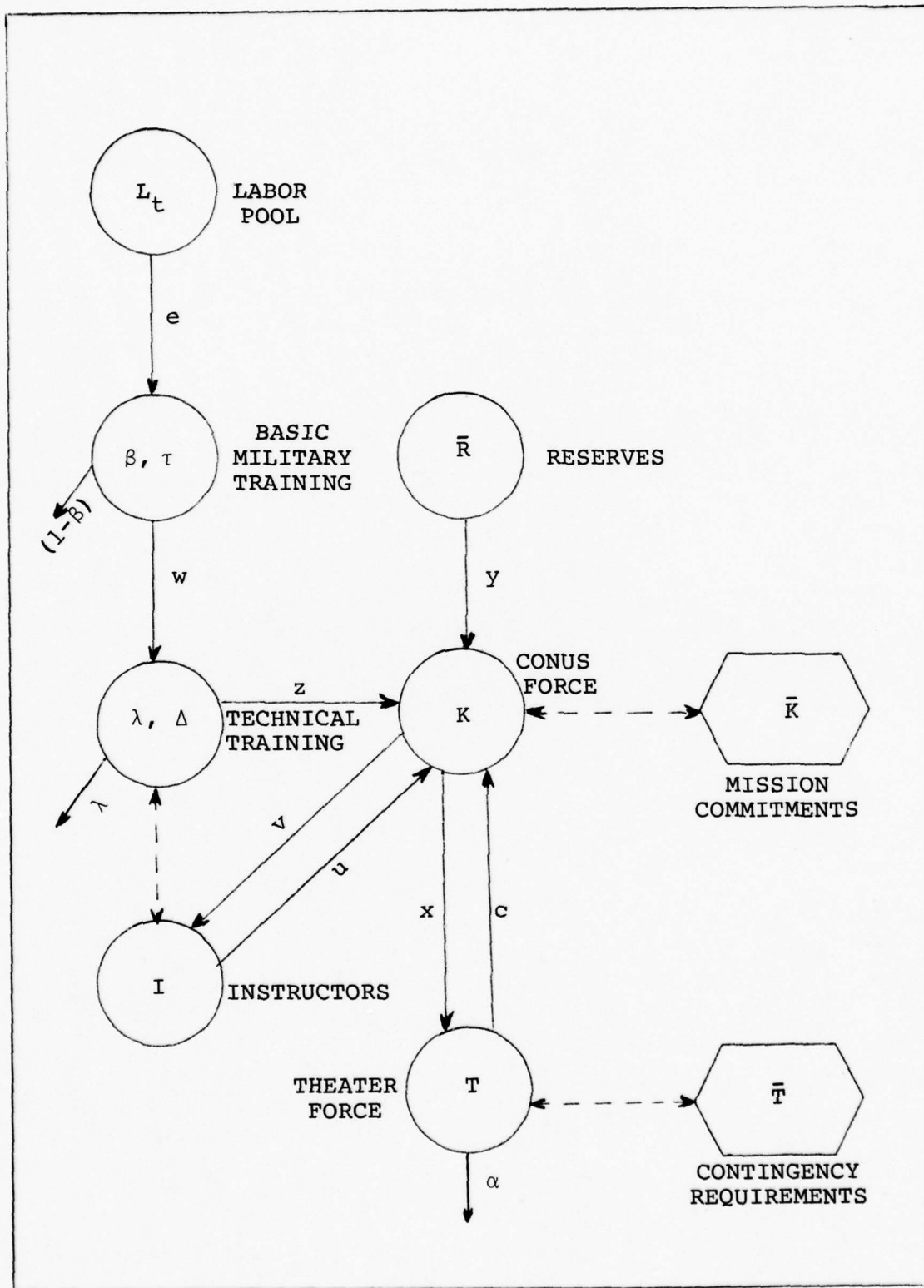


Figure 3-1. Contingency Planning Model

u - Flow rate of people from the instructor pool to the CONUS
 y - Flow rate of people from the reserves to CONUS
 z - Flow rate of people from technical training to the CONUS
 R - Number of people in the reserves
 \bar{R} - Desired residual reserve force size
 \bar{Y} - Activation limits on reserve forces
 Δ - Length of technical training
 λ - Attrition rate in technical training
 w - Flow rate of people into technical training
 b - Flow rate of people out of basic training
 e - Flow rate of people into basic training
 β - Survival rate in basic training
 τ - Length of basic training
 E - Desired trainee per instructor ratio
 L - Number of people in recruit supply pool
 I - Number of people in the instructor pool
 ρ - Maximum fraction of instructors in the instructor pool permitted to return to the CONUS force during the next time period

A basic assumption is that all transit times are less than one time interval.

Goal Equation Formulation

The system of goal equations for the model is now presented. Each equation set will be fully developed and

discussed prior to a presentation of the next equation set. Finally, the entire system of goal equations will be summarized.

People Available in the Overseas Theater. We want the people available in the overseas theater, $T_{i,t}$, to equal the requirements specified by $\bar{T}_{i,t}$ in each skill type i and time period t . This is formulated

$$T_{i,t} + d_{1,i,t}^- - d_{1,i,t}^+ = \bar{T}_{i,t} \quad (3.1)$$

However, the number of people currently in the theater force equals the number available at the previous time plus inflows minus outflows.

$$T_{i,t} = T_{i,t-1} + x_{i,t} - \alpha_{i,t} T_{i,t-1} - c_{i,t} \quad (3.2)$$

$\alpha_{i,t}$ is the combat attrition rate in the time interval $(t-1, t)$. Thus, we have

$$(1 - \alpha_{i,t}) T_{i,t-1} + x_{i,t} - c_{i,t} + d_{1,i,t}^- - d_{1,i,t}^+ = \bar{T}_{i,t} \\ \forall i; t = 2, 3, \dots, h \quad (3.3)$$

An initial value for $T_{i,0}$ is required for $t = 1$, and the resulting goal equation is expressed as

$$x_{i,1} - c_{i,1} + d_{1,i,1}^- - d_{1,i,1}^+ = \bar{T}_{i,t} \\ - T_{i,0}(1 - \alpha_{i,1}) \quad \forall i \quad (3.4)$$

In equation (3.2) the number of people currently overseas is expressed in terms of the number of people at the

last point in time plus subsequent inflows minus outflows. Likewise, the number of people currently overseas can be expressed in terms of the number of people initially overseas plus all subsequent inflows minus outflows. This is demonstrated below for skill type i at $t = 3$. Using equation (3.2) we find

$$\begin{aligned}
T_{i,3} &= T_{i,2} + x_{i,3} - \alpha_{i,3} T_{i,2} - c_{i,3} \\
T_{i,3} &= (1 - \alpha_{i,3}) T_{i,2} + (x_{i,3} - c_{i,3}) \\
T_{i,2} &= (1 - \alpha_{i,2}) T_{i,1} + (x_{i,2} - c_{i,2}) \\
T_{i,1} &= (1 - \alpha_{i,1}) T_{i,0} + (x_{i,1} - c_{i,1}) \\
T_{i,2} &= (1 - \alpha_{i,2}) (1 - \alpha_{i,1}) T_{i,0} \\
&\quad + (1 - \alpha_{i,2}) (x_{i,1} - c_{i,1}) + (x_{i,2} - c_{i,2}) \\
T_{i,3} &= (1 - \alpha_{i,3}) (1 - \alpha_{i,2}) (1 - \alpha_{i,1}) T_{i,0} \\
&\quad + (1 - \alpha_{i,3}) (1 - \alpha_{i,2}) (x_{i,1} - c_{i,1}) \\
&\quad + (1 - \alpha_{i,3}) (x_{i,2} - c_{i,2}) \\
&\quad + (x_{i,3} - c_{i,3})
\end{aligned} \tag{3.5}$$

In general, we can say for time $t \geq 4$

$$\begin{aligned}
T_{i,t} &= T_{i,0} \prod_{j=1}^t (1 - \alpha_{i,j}) \\
&\quad + (x_{i,1} - c_{i,1}) \prod_{j=2}^t (1 - \alpha_{i,j}) \\
&\quad + (x_{i,2} - c_{i,2}) \prod_{j=3}^t (1 - \alpha_{i,j}) + \dots \\
&\quad + (x_{i,t-1} - c_{i,t-1}) (1 - \alpha_{i,t}) \\
&\quad + (x_{i,t} - c_{i,t})
\end{aligned} \tag{3.6}$$

Therefore, the first set of goal equations is represented by

$$T_{i,t} + d_{1,i,t}^- - d_{1,i,t}^+ = \bar{T}_{i,t} \quad (3.7)$$

where $T_{i,t}$ is expressed in equations (3.5) and (3.6) as appropriate. The desired overseas goal $\bar{T}_{i,t}$ and the initial overseas strength $T_{i,0}$ are assumed to be known. If values for the combat attrition rate $\alpha_{i,t}$ are also assumed, then equation (3.7) represents a linear set of goal equations.

People Available in the CONUS Base. We desire also that the number of people in the CONUS base $K_{i,t}$ satisfy a requirement $\bar{K}_{i,t}$ that will fulfill the CONUS mission.

$$K_{i,t} + d_{2,i,t}^- - d_{2,i,t}^+ = \bar{K}_{i,t} \quad (3.8)$$

Again, the number of people on station at time t is the number on station at the last point in time plus subsequent net flows.

$$\begin{aligned} K_{i,t} = & K_{i,t-1} + z_{i,t} + c_{i,t} + u_{i,t} + y_{i,t} \\ & - x_{i,t} - v_{i,t} \end{aligned} \quad (3.9)$$

Thus, we have

$$\begin{aligned} z_{i,1} + c_{i,1} + u_{i,1} + y_{i,1} - x_{i,1} - v_{i,1} \\ + d_{2,i,1}^- - d_{2,i,1}^+ = \bar{K}_{i,1} - K_{i,0} \end{aligned} \quad (3.10)$$

$\forall i; t = 1$

and

$$\begin{aligned}
 K_{i,t-1} + z_{i,t} + c_{i,t} + u_{i,t} + y_{i,t} - x_{i,t} \\
 - v_{i,t} + d_{2,i,t}^- - d_{2,i,t}^+ = \bar{K}_{i,t} \quad (3.11) \\
 \forall i; t \geq 2
 \end{aligned}$$

As in the first goal equation set, the number of people currently in the CONUS can be expressed in terms of the number of people initially in the CONUS plus all subsequent net flows.

$$\begin{aligned}
 K_{i,t} = K_{i,0} + \sum_{j=1}^t (z_{i,j} + c_{i,j} + u_{i,j} \\
 + y_{i,j} - x_{i,j} - v_{i,j}) \quad (3.12)
 \end{aligned}$$

At the beginning of the contingency we assume that all normal personnel separations are terminated for the duration of hostilities. Therefore, no losses due to attrition in the CONUS are assumed.

The flow rate of people from technical training $z_{i,t}$ is determined by the flow rate of people into technical training and the technical training attrition rate.

$$z_{i,t} = (1 - \lambda_i) w_{i,t-\Delta_i} \quad (3.13)$$

λ_i is the attrition rate from technical training and Δ_i is the length of the technical training program. (Δ_i is an integer number of time periods.) Substituting the expression for $z_{i,t}$ above into equation (3.12), we find

$$K_{i,t} = K_{i,0} + (1 - \lambda_i) \sum_{j=1}^t w_{i,j-\Delta_i} + \sum_{j=1}^t (c_{i,j} + u_{i,j} + y_{i,j} - x_{i,j} - v_{i,j}) \quad (3.14)$$

Therefore, the second set of goal equations is represented by

$$K_{i,t} + d_{2,i,t}^- - d_{2,i,t}^+ = \bar{K}_{i,t} \quad (3.15)$$

where $K_{i,t}$ is expressed in equation (3.14) above. The desired CONUS goal $\bar{K}_{i,t}$ and the initial CONUS strength $K_{i,0}$ are assumed to be known. Additionally, the flow rates of people into technical training prior to the beginning of hostilities ($w_{i,1-\Delta_i}, w_{i,2-\Delta_i}, \dots, w_{i,0}$) and the technical training attrition rate λ_i are also assumed to be known.

Movement from the Reserve Forces. We want to activate the reserves as needed and assume that no new people are placed in the reserves for the duration of the contingency. Thus, there is an upper limit on the reserve total drawdown. The third set of goal equations is represented by

$$\sum_{t=1}^h y_{i,t} + d_{3,i}^- - d_{3,i}^+ = R_{i,0} - \bar{R}_i \quad (3.16)$$

where $R_{i,0}$ is the initial size of the reserve and \bar{R}_i is the desired residual reserve force size. We assume that both $R_{i,0}$ and \bar{R}_i are known.

We will also assume that there is a limitation on the number of reserves which can be activated in any given time period. The fourth set of goal equations is represented by

$$y_{i,t} + d_{4,i,t}^- - d_{4,i,t}^+ = \bar{y}_{i,t} \quad \forall i; t \quad (3.17)$$

where $\bar{y}_{i,t}$ is the activation limits on the reserves in skill level i during the time period $(t-1, t)$. We assume that $\bar{y}_{i,t}$ is known.

Inputs into Technical Training. The flow rate of people into technical training in the various skill levels is limited by the flow rate of people out of basic training.

$$\sum_{i=1}^n w_{i,t} + d_{5,t}^- - d_{5,t}^+ = b_{t-1} \quad (3.18)$$

(One time period delay is provided for administrative purposes and leave between basic training and technical training.) But

$$b_t = \beta e_{t-\tau} \quad (3.19)$$

where β is the basic training survival rate and e_t is the flow rate of people into basic training. τ is the length of basic training (an integral number of time periods). The fifth set of goal equations, therefore, is represented by

$$\sum_{i=1}^n w_{i,t} - \beta e_{t-\tau-1} + d_{5,t}^- - d_{5,t}^+ = 0 \quad \forall t \quad (3.20)$$

We assume that β and the flow rates of people into basic training prior to the beginning of hostilities ($e_{-\tau}, e_{1-\tau}, \dots, e_0$) are known.

Technical Training Instructors. We would like to maintain a fairly fixed ratio between the number of trainees and the number of instructors. This is represented by

$$w_{i,t} - E_i I_{i,t-1} + d_{6,i,t}^- - d_{6,i,t}^+ = 0 \quad (3.21)$$

where E_i is the desired trainee per instructor ratio for skill type i .

The number of instructors at time t equals the number at $t-1$ plus net flows.

$$I_{i,t} = I_{i,t-1} + v_{i,t} - u_{i,t} \quad (3.22)$$

Thus, we reformulate equation (3.21) in the following manner:

$$\begin{aligned} w_{i,t} - E_i I_{i,t-2} - E_i (v_{i,t-1} - u_{i,t-1}) \\ + d_{6,i,t}^- - d_{6,i,t}^+ = 0 \end{aligned} \quad (3.23)$$

Our sixth set of goal equations for the first time period is represented by

$$\begin{aligned} w_{i,1} - E_i I_{i,-1} - E_i (v_{i,0} - u_{i,0}) \\ + d_{6,i,1}^- - d_{6,i,1}^+ = 0 \end{aligned} \quad (3.24)$$

We assume that E_i , $I_{i,-1}$, $v_{i,0}$, and $u_{i,0}$ are all known.

We can also say that the number of instructors at time t equals the number initially plus net flows.

$$I_{i,t} = I_{i,0} + \sum_{j=1}^t (v_{i,j} - u_{i,j}) \quad (3.25)$$

Based on equation (3.25) we can reformulate equation (3.21) for $t > 1$ in the following manner:

$$\begin{aligned} w_{i,t} - E_i \sum_{j=1}^{t-1} (v_{i,j} - u_{i,j}) + d_{6,i,t}^- \\ - d_{6,i,t}^+ = E_i I_{i,0} \end{aligned} \quad (3.26)$$

Equation (3.26) represents our sixth set of goal equations for $t > 1$. We assume that $I_{i,0}$ is known.

Flow Into Basic Military Training. The flow rate of people into basic training is limited by the supply of recruits available at time t . The seventh set of goal equations is represented by

$$e_t + d_{7,t}^- - d_{7,t}^+ = L_t \quad \forall t \quad (3.27)$$

We assume that the supply of recruits L_t is known and is related to the timing on the activation of the draft system.

Flow of Instructors to CONUS. In order to provide stability to the instructor force, the flow rate of instructors from the instructor pool to the CONUS should not exceed some predesignated percentage (ρ) of instructors in the pool at any time t . This policy may be expressed by

$$u_{i,t} - \rho I_{i,t-1} + d_{8,i,t}^- - d_{u,i,t}^+ = 0 \quad (3.28)$$

The eighth set of goal equations for $t=1$ is represented by

$$u_{i,1} + d_{8,i,1}^- - d_{8,i,1}^+ = \rho I_{i,0} \quad (3.29)$$

By using equation (3.25) we can derive an expression for $I_{i,t-1}$ in terms of $I_{i,0}$ and net instructor flows.

$$I_{i,t-1} = I_{i,0} + \sum_{j=1}^{t-1} (v_{i,j} - u_{i,j}) \quad (3.30)$$

Combining equation (3.30) with equation (3.28), we find the eighth set of goal equations for $t > 1$.

$$u_{i,t} - \rho \sum_{j=1}^{t-1} (v_{i,j} - u_{i,j}) + d_{8,i,t}^- - d_{8,i,t}^+ = \rho I_{i,0} \quad (3.31)$$

Summary of Goal Equations

The eight sets of goal equations are summarized below. Terms containing only constants (involving initial conditions and assumed values for certain parameters) have been combined with the righthand-side goal values of the equations. This action is necessary in order to make the goal equations conformable to a simplex solution procedure.

First Equation Set--People Available in the Overseas Theater.

$$\begin{aligned} x_{i,1} - c_{i,1} + d_{1,i,1}^- - d_{1,i,1}^+ \\ = \bar{T}_{i,1} - T_{i,0} (1 - \alpha_{i,1}) \\ \forall i; t = 1 \end{aligned} \quad (3.32)$$

$$\begin{aligned}
& (x_{i,2} - c_{i,2}) + (1 - \alpha_{i,2}) (x_{i,1} - c_{i,1}) \\
& + d_{1,i,2}^- - d_{1,i,2}^+ = \bar{T}_{i,2} - T_{i,0} (1 - \alpha_{i,1}) \\
& \cdot (1 - \alpha_{i,2}) \quad \forall i; t = 2 \quad (3.33)
\end{aligned}$$

$$\begin{aligned}
& (x_{i,3} - c_{i,3}) + (1 - \alpha_{i,3}) (x_{i,2} - c_{i,2}) \\
& + (1 - \alpha_{i,3}) (1 - \alpha_{i,2}) (x_{i,1} - c_{i,1}) + d_{1,i,3}^- \\
& - d_{1,i,3}^+ = \bar{T}_{i,3} - T_{i,0} (1 - \alpha_{i,1}) (1 - \alpha_{i,2}) \\
& \cdot (1 - \alpha_{i,3}) \quad \forall i; t = 3 \quad (3.34)
\end{aligned}$$

$$\begin{aligned}
& (x_{i,t} - c_{i,t}) + (x_{i,t-1} - c_{i,t-1}) (1 - \alpha_{i,t}) + \dots + \\
& (x_{i,2} - c_{i,2}) \prod_{j=3}^t (1 - \alpha_{i,j}) + (x_{i,1} - c_{i,1}) \\
& \cdot \prod_{j=2}^t (1 - \alpha_{i,j}) + d_{1,i,t}^- - d_{1,i,t}^+ \\
& = \bar{T}_{i,t} - T_{i,0} \prod_{j=1}^t (1 - \alpha_{i,j}) \quad \forall i; t \geq 4 \quad (3.35)
\end{aligned}$$

Second Equation Set--People Available in the CONUS

Base.

$$\begin{aligned}
& \sum_{j=1}^t (c_{i,j} + u_{i,j} + y_{i,j} - x_{i,j} - v_{i,j}) \\
& + d_{2,i,t}^- - d_{2,i,t}^+ = \bar{K}_{i,t} - K_{i,0} \\
& - (1 - \lambda_i) \sum_{j=1}^t x_{i,j-\Delta_i} \quad \forall i; t \leq \Delta_i \quad (3.36)
\end{aligned}$$

$$\begin{aligned}
& (1 - \lambda_i) \sum_{j=\Delta_i+1}^t w_{i,j-\Delta_i} \\
& + \sum_{j=1}^t (c_{i,j} + u_{i,j} + y_{i,j} - x_{i,j} - v_{i,j})
\end{aligned}$$

$$\begin{aligned}
& + d_{2,i,t}^- - d_{2,i,t}^+ = \bar{K}_{i,t} - K_{i,0} \\
& - (1 - \lambda_i) \sum_{j=1}^{\Delta_i} w_{i,j-\Delta_i} \quad \forall i; t > \Delta_i \quad (3.37)
\end{aligned}$$

Third Equation Set--Movement From the Reserves:

Reserves Total Drawdown.

$$\begin{aligned}
& \sum_{t=1}^h y_{i,t} + d_{3,i}^- - d_{3,i}^+ = R_{i,0} - \bar{R}_i \\
& \quad \quad \quad \forall i \quad (3.38)
\end{aligned}$$

Fourth Equation Set--Movement From the Reserves:

Reserve Activation Limits.

$$y_{i,t} + d_{4,i,t}^- - d_{4,i,t}^+ = \bar{y}_{i,t} \quad \forall i, t \quad (3.39)$$

Fifth Equation Set--Inputs Into Technical Training.

$$\begin{aligned}
& \sum_{i=1}^n w_{i,t} + d_{5,t}^- - d_{5,t}^+ \\
& = \beta e_{t-\tau-1} \quad t \leq \tau + 1 \quad (3.40)
\end{aligned}$$

$$\begin{aligned}
& \sum_{i=1}^n w_{i,t} - \beta e_{t-\tau-1} + d_{5,t}^- - d_{5,t}^+ = 0 \\
& \quad \quad \quad t > \tau + 1 \quad (3.41)
\end{aligned}$$

Sixth Equation Set--Technical Training Instructors.

$$\begin{aligned}
& w_{i,1} + d_{6,i,1}^- - d_{6,i,1}^+ \\
& = E_i I_{i,-1} - E_i (v_{i,0} - u_{i,0}) \quad \forall i; t = 1 \quad (3.42)
\end{aligned}$$

$$\begin{aligned}
& w_{i,t} - E_i \sum_{j=1}^{t-1} (v_{i,j} - u_{i,j}) + d_{6,i,t}^- \\
& - d_{6,i,t}^+ = E_i I_{i,0} \quad \forall i; t > 1 \quad (3.43)
\end{aligned}$$

Seventh Equation Set--Flow into Basic Military Training.

$$e_t + d_{7,t}^- - d_{7,t}^+ = L_t \quad \forall t \quad (3.44)$$

Eighth Equation Set--Flow of Instructors to CONUS.

$$\begin{aligned}
& u_{i,1} + d_{8,i,1}^- - d_{8,i,1}^+ = \rho I_{i,0} \\
& \quad \forall i; t = 1 \quad (3.45)
\end{aligned}$$

$$\begin{aligned}
& u_{i,t} - \rho \sum_{j=1}^{t-1} (v_{i,j} - u_{i,j}) + d_{8,i,t}^- \\
& - d_{8,i,t}^+ = \rho I_{i,0} \quad \forall i; t > 1 \quad (3.46)
\end{aligned}$$

Objective Function Formulation

The objective function is constructed from deviation variables relating to goals in various preemptive priority levels. Because of the complexity of the contingency planning problem, all goals as specified in the system of goal equations normally cannot be fully satisfied. Therefore, we must establish a hierarchy of importance among goals and express this hierarchy in terms of a preemptive priority structure.

Prior to establishing a preemptive priority structure and formulating the objective function, some discussion of our contingency planning problem is warranted. The

purpose of our model is to assist us in planning to meet overseas contingency requirements. Therefore, we should assign high priorities to goals associated with meeting required overseas personnel levels. At the same time, however, we should also consider satisfying mission commitments for personnel in the CONUS. Failure to meet CONUS commitments adequately could result in a possible degradation of strategic preparedness. Thus, we should also assign high priorities to goals associated with meeting required CONUS personnel levels. Similarly, a strong argument can be made for assuring that an adequate instructor pool be provided for the large input of trainees expected with the anticipated initiation of the draft system. Finally, certain goals within the problem relate to resource limitations which are normally outside of the control of a decision maker. There is a limit on the availability of both the civilian labor pool and the reserve force which cannot be exceeded.

While all of the goals mentioned above are important, it is unrealistic to assume that all can be simultaneously satisfied. It is also unrealistic to pick one goal (such as the achievement of overseas personnel requirements) and use that goal in the formulation of a linear programming problem. Thus, we will establish our goals in a hierarchical order and use a preemptive priority structure in preparing our objective function. The hierarchy of goals reflects the needs and desires of a decision maker

at the time he formulates the problem. Two different decision makers (or the same decision maker formulating the problem at different times) could establish quite different goal hierarchies. A distinct advantage of goal programming is the fact that a decision maker may want to compare the overall satisfaction of goals under several different goal hierarchies. Thus, the hierarchy of goals and preemptive priority structure which we establish below represents just one of several reasonable possibilities.

First Priority Level. Our first priority level goals are concerned with resource limitations which are normally outside of the control of the decision maker. These goals have to be achieved, and, therefore, we need assign no weighting to the various deviation variables in the first priority level. Since we are concerned with not exceeding certain resource limitations, we utilize the positive deviation variables found in the appropriate resource goal equations. Specifically, we include positive deviation variables from the third goal equation set (Reserves Total Drawdown), fourth goal equation set (Reserves Activation Limits), and seventh goal equation set (Flow into Basic Military Training) in the first priority level. The flow rate of people into technical training is determined directly from the flow rate of people into basic training and the attrition rate in basic training. Therefore, we include both the positive and negative deviation variables

from the fifth goal equation set (Inputs into Technical Training) in the first priority level.

We also include in the first priority level our concern for maintaining a stable instructor pool by limiting the percentage of instructors who may be returned to the CONUS force during any time period. We include the positive deviation variables from the eighth goal equation set (Flow of Instructors to CONUS) in the first priority level.

Finally, we express a desire for a stable force structure after the contingency is concluded. In the latter periods of the contingency, the flow rate of people into basic training does not occur in time to help satisfy CONUS or overseas contingency requirements due to delays required for basic and technical training. However, we would like to maintain a stable force structure after the contingency in order to help satisfy any unforeseen future requirements. Therefore, we include negative deviation variables for the appropriate latter time periods from the seventh goal equation set (Flow into Basic Military Training) in the first priority level.

Second Priority Level. Our second priority level goals are concerned with providing an adequate instructor force to meet the needs of the large numbers of trainees expected to support our contingency. Our main concern is that our desired trainee per instructor ratio not be exceeded. Thus, we include the positive deviation

variables from the sixth goal equation set (Technical Training Instructors) in the second priority level. Additionally, we assign weights to those deviation variables for the particular skill levels and time periods that we favor over other skill levels and time periods. We may use unit weights if we have no such preferences.

It is logical to place the meeting of the desired trainee per instructor ratio in a higher priority level than the meeting of overseas and CONUS requirements for at least two reasons. First, personnel who enter the military service to help offset overseas and CONUS requirements need adequate training. Second, the size of an adequately manned instructor force is small compared to the overall CONUS and overseas force levels. Moreover, each instructor is capable of training several trainees. Therefore, small reductions in the CONUS force initially in order to build an adequate instructor pool will result in very significant gains in meeting overseas and CONUS requirements later in the contingency.

Third Priority Level. Our third priority level goals are concerned with meeting overseas theater personnel requirements. We would like to satisfy all of our overseas theater requirements. Thus, we include the negative deviation variables from the first goal equation set (People Available in the Overseas Theater) in the third priority level. In addition, we include a weighting structure

which reflects our preferences among the various skill levels and time periods.

Fourth Priority Level. Our fourth priority level goals are concerned with satisfying all of our CONUS personnel requirements. We include the negative deviation variables from the second goal equation set (People Available in the CONUS Base) in the fourth priority level. We include a weighting structure reflecting our preferences within the fourth priority level.

Fifth Priority Level. Our fifth priority level goals are very similar to our third priority level goals except that now we desire not to exceed our overseas theater personnel requirements. Thus, we include the positive deviation variables from the first goal equation set (People Available in the Overseas Theater) and a weighting structure reflecting our preferences in the fifth priority level.

Sixth Priority Level. Our final priority level is concerned with not exceeding our CONUS personnel requirements. We include the positive deviation variables from the second goal equation set (People Available in the CONUS Base) and a weighting structure reflecting our preferences in the sixth priority level.

Preemptive Goal Structure Summarized
in Objective Function

We will now incorporate our goals assigned in the six preemptive priority levels into an objective function. Since our various levels represent an ordinal ranking among

goals, we should not combine objective function terms from one level with terms from a different level. Thus, it is convenient to think of the objective function as a vector \bar{a} which contains one component a_i for each preemptive priority level P_i (Ref 15:16-17). With six preemptive priority levels, our objective function has the following form:

$$\bar{a} = (a_1, a_2, a_3, a_4, a_5, a_6) \quad (3.47)$$

Each term in the objective function consists of a summation of applicable deviation variables multiplied by their representative weights. As a result, we can easily identify the extent of goal satisfaction within each priority level. (For example, $a_i = 0$ implies that all goals at the i^{th} priority level have been completely satisfied.) Based on our earlier formulation of six preemptive priority levels, we now construct our objective function.

$$\bar{a} = (a_1, a_2, a_3, a_4, a_5, a_6) \quad (3.47)$$

where

$$\begin{aligned} a_1 = & \sum_{i=1}^n d_{3,i}^+ + \sum_{t=1}^h \sum_{i=1}^n d_{4,i,t}^+ \\ & + \sum_{t=1}^h d_{7,t}^+ + \sum_{t=1}^h (d_{5,t}^+ + d_{5,t}^-) \\ & + \sum_{t=1}^h \sum_{i=1}^n d_{8,i,t}^+ + \sum_{t=t_L}^h d_{7,t}^- \end{aligned} \quad (3.48)$$

$$a_2 = \sum_{t=1}^h \sum_{i=1}^n w_{6,i,t}^+ d_{6,i,t}^+ \quad (3.49)$$

$$a_3 = \sum_{t=1}^h \sum_{i=1}^n w_{1,i,t}^- d_{1,i,t}^- \quad (3.50)$$

$$a_4 = \sum_{t=1}^h \sum_{i=1}^n w_{2,i,t}^- d_{2,i,t}^- \quad (3.51)$$

$$a_5 = \sum_{t=1}^h \sum_{i=1}^n w_{1,i,t}^+ d_{1,i,t}^+ \quad (3.52)$$

$$a_6 = \sum_{t=1}^h \sum_{i=1}^n w_{2,i,t}^+ d_{2,i,t}^+ \quad (3.53)$$

and

$$t_L = h - \min (\Delta_i + \tau); i = 1, \dots, n$$

(t_L is the first time period in which gains in basic training can no longer influence the outcome of the contingency.)

$w_{k,i,t}^+$ and $w_{k,i,t}^-$ are the weights assigned to the positive and negative deviation variables, respectively.

Final Linear Goal Programming Model

Our final model consists of the objective function (equations (3.47) through (3.53)), the goal equations (equations (3.32) through (3.46)), and the requirement that each deviation variable be nonnegative ($x_{i,t}, c_{i,t}, v_{i,t}, y_{i,t}, u_{i,t}, w_{i,t}, e_t \geq 0, \forall i, t$).

First Numerical Example

In the last part of this chapter, we present a small numerical example for the model involving three skill levels and six time periods. Since we have already developed the general forms of the goal equations and objective function, we need only provide the specific numerical values for the various goals, initial conditions, objective function weights, and parameters of the model. The solution of our example problem is deferred until Chapter Four in which we discuss the simplex computer algorithm.

Our example problem concerns the assignment of enlisted personnel in three skill levels for six time periods. Each time interval $(t-1, t)$ is one month, and, thus our overall contingency lasts for six months. Our three skill levels are identified below.

i	Enlisted Personnel Category
1	Mission
2	Direct
3	Indirect Mission Support

Our initial CONUS force size (not including trainees or instructors) is provided below (in thousands of personnel).

i	1	2	3
$K_{i,0}$	75	152	170

i	1	2	3
Δ_i	1	2	1
λ_i	0	0	0

(Cross leveling could be handled by using a technical training attrition rate matrix Λ allowing flows of personnel among jobs and out of the system. Minor modifications to the second goal equation set would be required.)

The initial technical training pipeline consists of people who had entered technical training prior to the contingency but have not yet entered the CONUS force by the beginning of the contingency.

i	$w_{i,-1}$	$w_{i,0}$
1	---	1.67
2	3.17	3.17
3	---	3.33

Data related to desired trainee per instructor ratio and initial instructor levels and flows is provided below.

i	E_i	$I_{i,-1}$	$I_{i,0}$	$v_{i,0}$	$u_{i,0}$
1	8	.209	.209	0	0
2	5	.634	.634	0	0
3	10	.333	.333	0	0

We desire that the CONUS force remain at initial peacetime levels throughout the contingency.

i	1	2	3
$\bar{K}_{i,t} \quad t > 0$	75	152	170

The length of basic training is assumed to be one month ($\tau=1$) due to the small size of our problem. The basic training survival rate is assumed to be 70 percent ($\beta=.7$). The initial basic training pipeline consists of people who had entered basic training prior to the contingency, but have not yet entered technical training by the beginning of the contingency. (Entry levels provide replacements necessary to support a normal peacetime force turnover rate of once every five years.)

e_{-1}	e_0
11.67	11.67

The length of technical training is dependent upon the skill level. We assume no attrition in technical training in order to avoid the need for cross leveling of personnel among skills.

We desire that no more than fifty percent of our on-hand instructors be returned to the CONUS force during any time period ($\rho = .5$).

We assume that our initial reserves are equally balanced for the three skill levels and that no residual reserve forces are to be maintained. (Units are thousands of personnel.)

i	$R_{i,0}$	\bar{R}_i
1	30	0
2	30	0
3	30	0

We also assume that up to tenthousand available reserves can be activated each month in each skill level ($\bar{Y}_{i,t} = 10 \forall i,t$).

During the first month of the contingency, only peacetime levels of recruits are available. The draft system is initiated by the beginning of the second month, the labor availability increases accordingly. (Units are thousands of personnel.)

t	L_t
1	11.67
≥ 2	50

Our initial overseas theater force is at its peace-time levels. (Units are thousands of personnel.)

i	1	2	3
$T_{i,0}$	25	38	30

However, our contingency requires a rapid growth in overseas levels such that our total combined CONUS and overseas force level increases by approximately fifty percent by the end of the contingency. (Units are thousands of personnel.)

Overseas Force Goals, $\bar{T}_{i,t}$

$\begin{matrix} i \\ t \end{matrix}$	1	2	3
1	29	42	32
2	33	60	35
3	50	83	40
4	70	113	45
5	85	138	60
6	95	165	75

Finally, our monthly overseas attrition rates depend upon both the applicable time periods and skill levels. We assume that losses are fairly significant during the initial months of the contingency.

Monthly Combat Attrition Rates, $\alpha_{i,t}$

$\begin{smallmatrix} i \\ t \end{smallmatrix}$	1	2	3
1	.03	.02	.02
2	.03	.02	.02
3	.02	.013	.01
4	.01	.005	.001
5	.01	.005	.001
6	.01	.005	.001

We have now established all of the numerical values needed to solve our problem except for the weights in the objective function. Presently, unit values are assigned to all weights to reflect no special preferences among the various skill levels and time periods. We may modify the weights later.

Our present example problem consists of 114 decision variables and 105 goal equations with 150 deviation variable terms in the objective function. This problem and larger problems are solved in Chapter Four.

IV. Development and Testing of the Simplex Algorithm

Initial Steps

The simplex method of mathematical programming is an effective analytical tool for determining the optimal value of an objective function given a set of constraints. In a goal programming context, the objective function is a vector quantity with more than one measure of effectiveness. The simplex method for linear goal programming iterates progressively on one priority level of the objective function (sometimes referred to as the achievement function) at a time. This method switches to the next highest priority level when improvement is no longer possible at the current level.

Although a variety of simplex computer codes is available for linear programming, only three explicit codes were found in the literature for the linear goal programming context. The code in the Marini thesis illustrates a multiphase version of linear programming in which each priority level is treated as a separate phase in the execution process. Marini tries to solve goal programming problems with a linear programming approach--each priority level is solved as a distinct linear programming simplex problem. The automated nature of his program allows every

level to be handled during one run without manual initiation of each phase (Ref 20). The other two codes from Lee and Ignizio (Refs 19; 15) are written in a straightforward manner and have been tested for small size samples. Because of their ease of adaptability, simplicity, and flexibility, these latter two codes appear attractive for the analysis of the contingency planning model.

One criterion that we consider in dealing with the contingency model is the number of goal equations used to describe the problem. One aim of our model is to experiment with the sizes of military forces in terms of physical numbers of individuals and of different job skills flowing over several time periods in the contingency. We want to determine how effectively a personnel force configuration can support a given war scenario; we want to determine what size of force is necessary to handle the war scenario and what size of war (length of time, attrition factors) make our capabilities with human resources a limiting factor. In order to accommodate more and more job categories and time periods, the computer code needs the capability to expand and store more variables and goal equations.

Ignizio's computer code, a modified simplex procedure, demonstrates obvious savings in storage capacity over Lee's algorithm. Both codes deal with the same technology matrix; Ignizio excludes the righthand-side submatrix of negative deviation variable coefficients; Ignizio's algorithm pivots on the remaining technology coefficients and places the

departing basic variable of a particular row into the column of the variable which enters the basis. This switching of variables between the pivoting row and column avoids a duplication of variables in the basis and in the column headings. Also, Ignizio's code provides helpful subprograms which Lee's does not--subprograms to generate alternate optimal solutions and to fix numbers as integers if requested by the user. (This latter subprogram changes numerical values which stray from remaining as integers into integers if the values lie within a constant tolerance of that integer. Precision of variables and limited control of roundoff error is retained.) Because of its advantages of storage requirements and its useful subprograms, the computer code in Ignizio's textbook serves as the foundation of the goal programming simplex for our contingency model.

Figure 4-1 shows Ignizio's original design; the functional description of each subprogram follows:

GMAIN reads in all input data and controls the execution phase of the algorithm; appropriate errors are printed.

PLACE assembles the initial objective function weights in the matrices of top stub and left stub values. In Ignizio's context, these matrices perform the same functions in the simplex method as does the c_j top row of Lee's model (c_j row defined in Chapter Two).

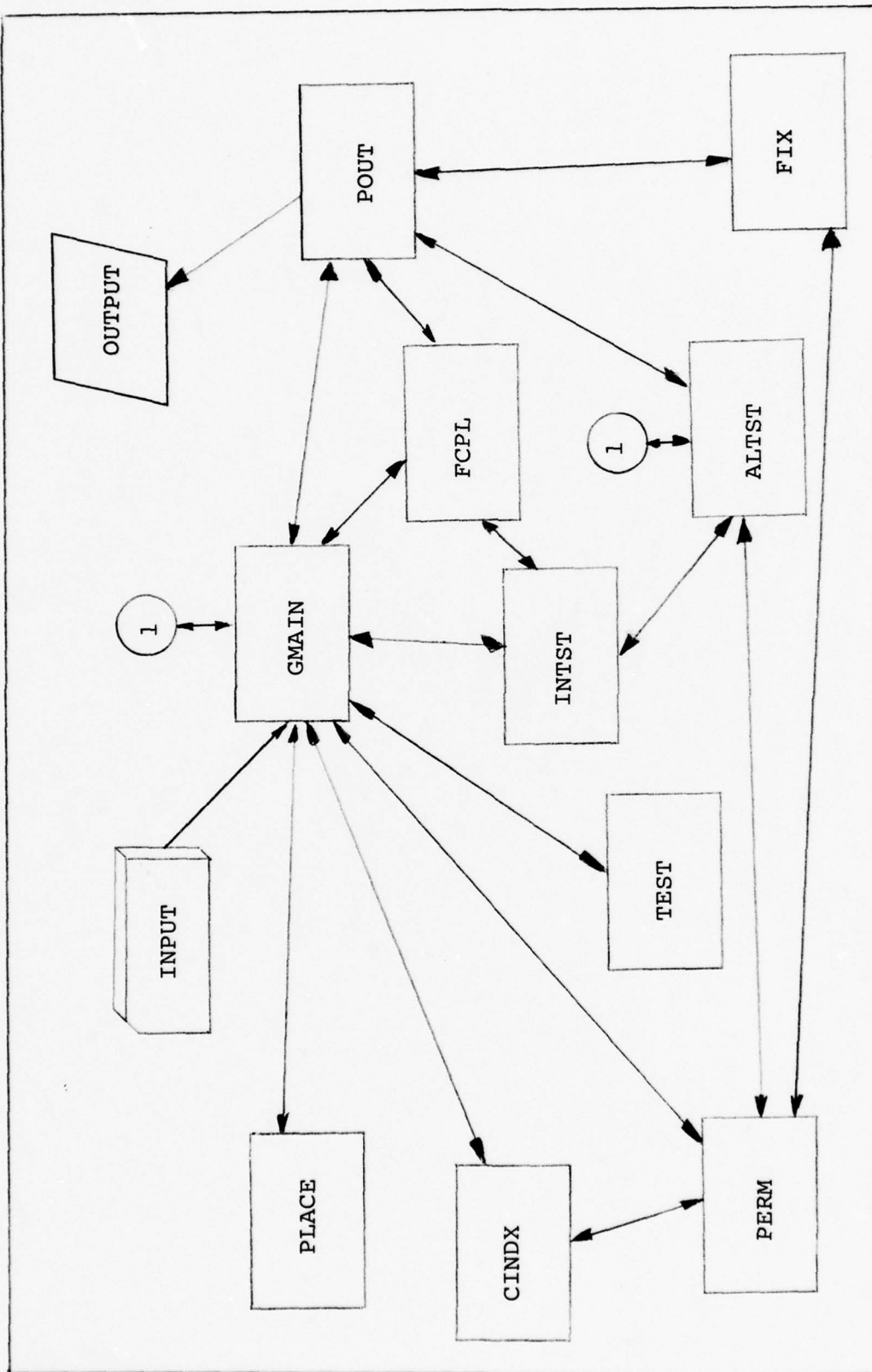
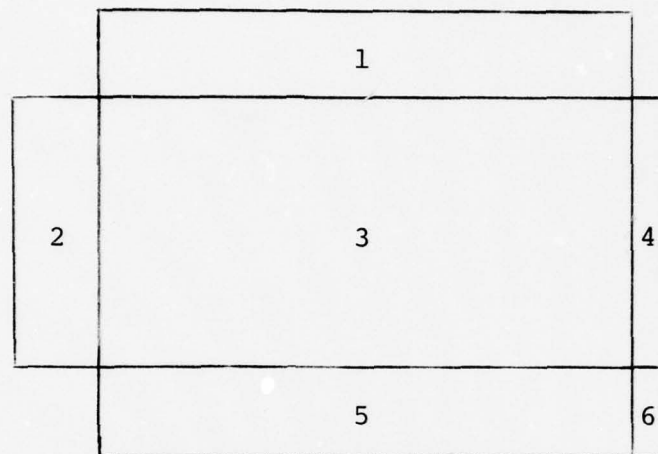


Figure 4-1. Original Design

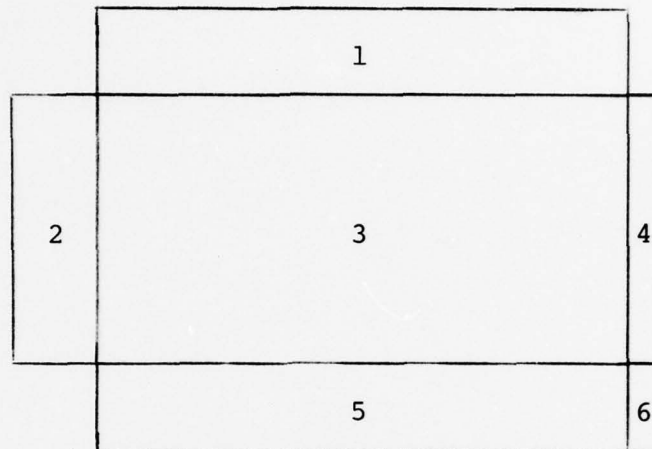


Basic Tableau Setup

Legend

1. Top stub values (c_j top rows)
2. Left stub values (similar to top stub values)
3. Technology matrix of goal equation coefficients (TE)
4. Righthand-side goal values
5. Index rows ($z_j - c_j$ matrix)
6. Objective function levels of achievement

Figure 4-1. Continued



Basic Tableau Setup

Legend

1. Top stub values (c_j top rows)
2. Left stub values (similar to top stub values)
3. Technology matrix of goal equation coefficients (TE)
4. Righthand-side goal values
5. Index rows ($z_j - c_j$ matrix)
6. Objective function levels of achievement

Figure 4-1. Continued

CINDX computes new index rows of the $(z_j - c_j)$ matrix and the latest values of the objective function components.

TEST determines the departing variable row and the entering variable column by means of the new index row values.

PERM pivots the technology matrix on the element of intersection of the column and row which are determined in TEST; values are changed not only throughout the technology matrix but also in the matrix of righthand-side goal values through this permutation.

FIX retains the precision of integer-valued quantities.

POUT prints the processed information of each optimal solution.

ALTST tests the final technology matrix for additional alternative solutions if any exist.

FCPL forms a cutting plane construct for the problem and adds the necessary cutting plane equations to the tableau (technology matrix) whenever an integer solution is requested.

INTST indicates whether or not a requested variable is integer-valued (Ref 15:227-228, 234-242).

The arrows in the figure represent the lines of communication between the subprograms and between the input/output media and the execution program. (The original

design and updated design are created in the language of FORTRAN IV Extended, and the discussion which follows assumes the reader to have a very basic knowledge of the language.)

For our contingency model, we decided early in the algorithm development stages to exclude those subprograms dealing with integer-valued solutions. Considering military forces, the problem that we are modelling deals with thousands of personnel, and a scaling of the problem to fit "units" of a thousand men in both input and output results. Optimal values to the nearest person are available when the problem is scaled. Thus, FCPL and INTST are dropped from the model as well as the initial input control variable that specifies that an integer-valued solution be printed. Only program FIX is retained for its ability to help control the effects of roundoff error.

The original design is fundamentally modular, for each subroutine performs one and only one specific function of the algorithm. It is unnecessary, though, to have the main program read the initial, unprocessed data and write error diagnostics; instead, other subroutines are constructed to handle this input/output processing and still retain a modular structure. The addition of these subroutines makes a clear distinction between the initialization phase and the execution phase of the computer algorithm. Furthermore, the main program now acts like an executive module, only controlling the program execution. If difficulties

in processing the input and the output arise, the main program is alerted and halts the process so that the user may correct the error.

The initialization phase consists of the newly created routines INIT and COEF and the subroutine PLACE, whose function remains unchanged from the original design. INIT directs the reading of input data so that, at INIT's conclusion, every initial matrix value is in place, and the algorithm is ready to test for optimality of the system at the first priority level. Labels of twenty characters maximum for every decision variable are inserted into the model prior to INIT's conclusion. INIT resembles a miniature executive module that does minimal input processing and directs the initialization through subroutine calls.

COEF generates initial coefficients of the goal equations and inserts them into proper positions of the technology matrix (referred to as TE in the code). INIT is responsible for reading the initial righthand-side values, but COEF is responsible for generating updated values for the righthand-sides of the goal equations. The original design reads in the TE coefficients one row at a time with formatted READ statements. Reducing input/output time in the updated design is effected by the calculation of coefficients with the initial data such as $T_{i,0}$, $K_{i,0}$, and β which were explained in the previous chapter. The algorithm requires a sparse TE matrix (with few nonzero elements) where coefficients of earlier time periods of

the contingency help to build the coefficients of subsequent time periods; COEF performs this arithmetic.

The original design reads objective function values also as combinations of the priority level, the particular deviation variable subscript, and the weighting factor, with each combination on a single card. Input processing time is reduced even further in the updated design. Our model allows these input values to be read in as four combinations maximum of objective function values on a single card, and, hence, requires fewer READ's to the input file and one-quarter as many physical cards.

The execution phase of the algorithm consists of the originally designed CINDX, TEST, PERM, POUT, and ALTST (with slight modifications for efficient code) with an updated version of FIX and the addition of subroutine ERRORS. ERRORS is called by the MAIN program only if error diagnostics are needed. Once either ALTST or ERRORS ends, the algorithm is complete with the current problem, and another is begun.

FIX is changed to permit a variable number L of decimal places of precision. L is an integer greater than one. In the original design FIX assured precision to the fourth decimal place by examining the input value and confirming whether or not it lies in an interval of .005 from an integer. With the modification, FIX examines the input value and determines if it lies within a distance of $5 \times 10^{-(L-1)}$ from an integer. If so, the integer value

is returned in the function; otherwise, the original value is returned unchanged.

The design and logic flow of the updated model are depicted in Figure 4-2. The algorithm begins with INIT directing the initialization phase. Objective function values are in place as are technology equation coefficients. Control returns to the MAIN program with the variable NROW set to one. NROW serves as the indicator of the current priority level at which the program is operating.

CINDX creates the index rows which are tested later with the technology equations for the entering variables. The criterion to select the entering variable is aptly stated by Ignizio: "Select the largest, positive $I_{k,s}$ for which there are no negative valued index numbers, at a higher priority, in the same column." ($I_{k,s}$ in the citation is the value in the index matrix at the k^{th} priority level (row) and the s^{th} column (Ref 15:47).) The departing variable is then determined by finding the minimum value of the quotients $b_i/e_{i,s}$ where b_i is the righthand-side goal equation quantity and $e_{i,s}$ is the corresponding i^{th} row element of the entering variable column. This determination of variables of the model is one iteration of the algorithm at the current priority level.

If the entering column number is calculated to be zero, the priority level cannot be optimized further, and control passes to the next priority level. Otherwise, permutation of the TE matrix followed by another call to

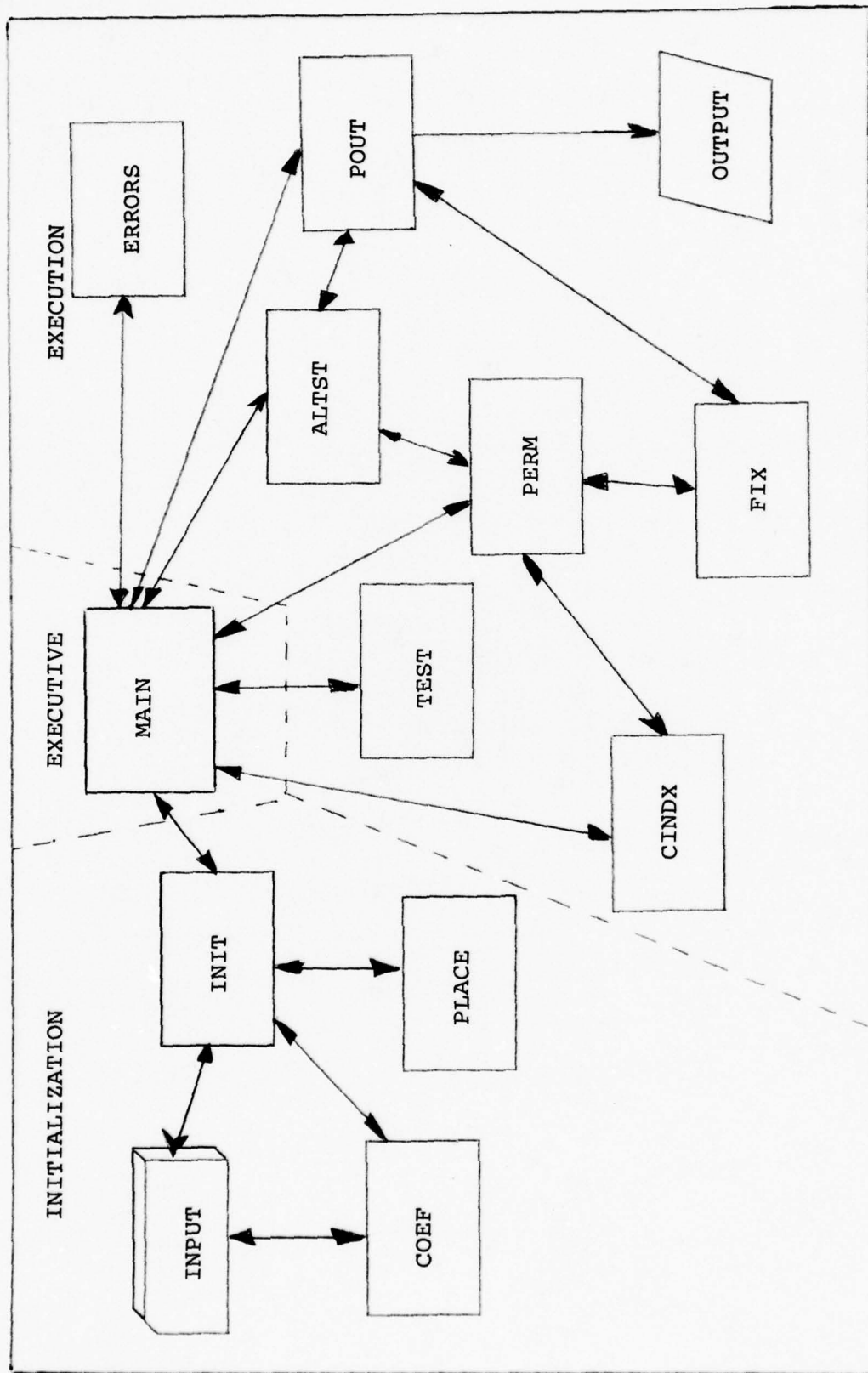


Figure 4-2. Updated Model: Modular Design and Logic Flow

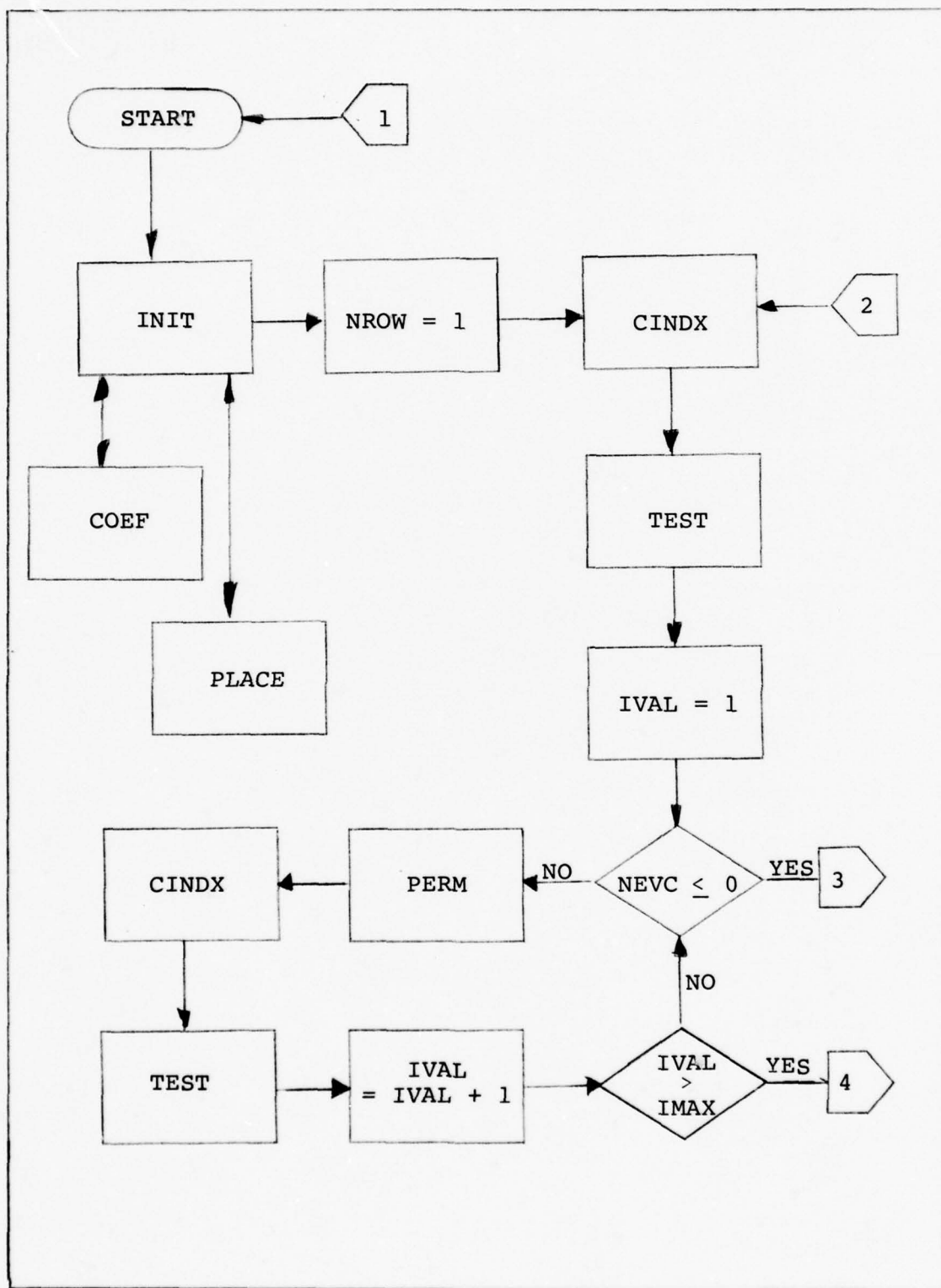


Figure 4-2--Continued

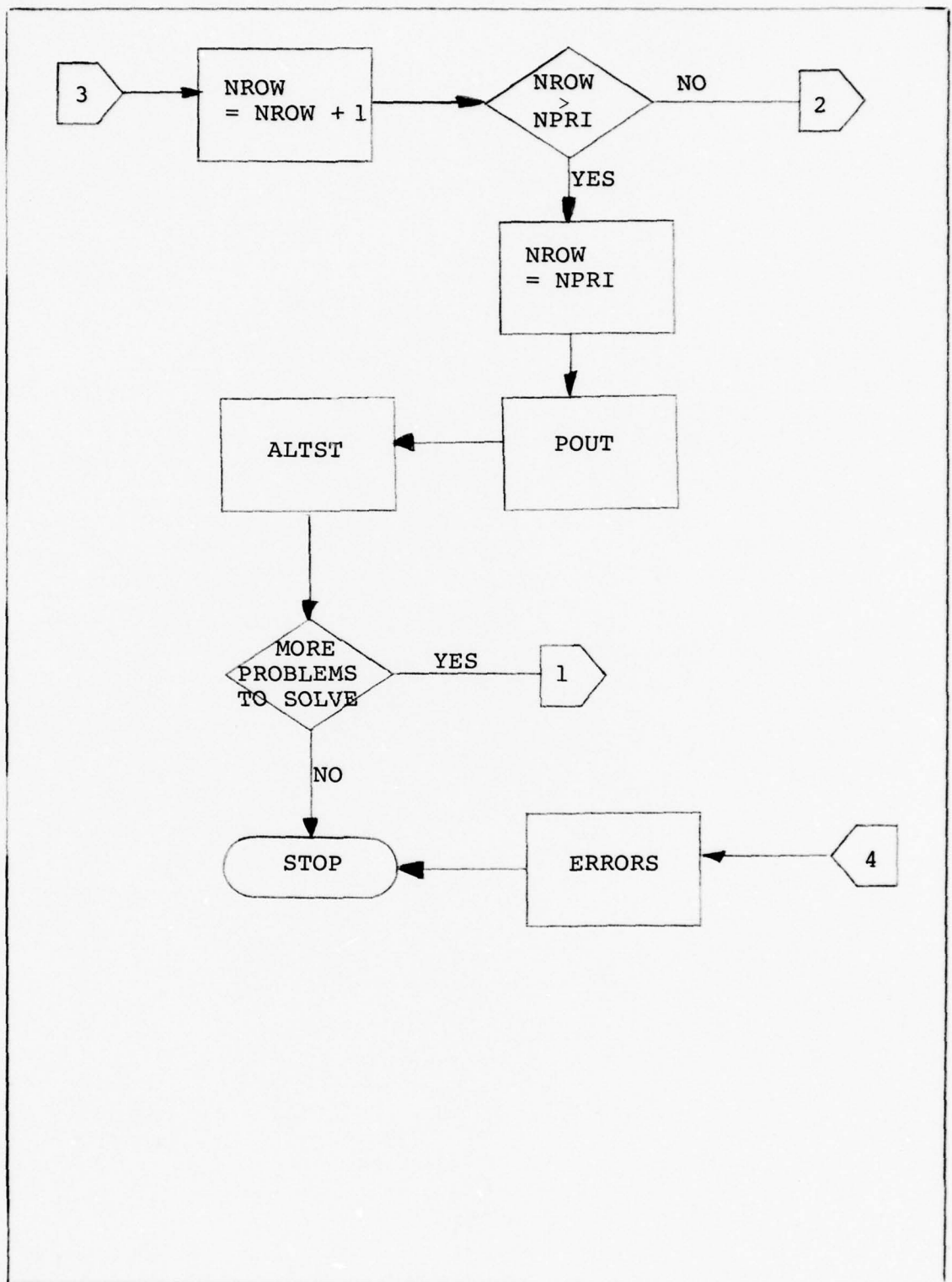


Figure 4-2--Continued

CINDEX prepares the process to iterate once more to determine the principal row and column. This iterative process continues for each priority level until one of two conditions is satisfied: either the column value is zero or the total number of iterations exceeds an initial number called IMAX. (IMAX for our problem is 200.) If the first condition is satisfied, the priority level NROW is incremented by one, and the process continues until NROW exceeds the number of priority levels in the objective function. When NROW does exceed that number, the execution phase becomes one of printing optimal solutions (the first optimal and then the alternate optimal solutions) and of stopping unless another problem is ready in the data stream. When the number of iterations exceeds IMAX, subroutine ERRORS has control over execution, prints the error statement, and returns control to MAIN for additional input contingency problems.

The design of COEF requires further description. This subprogram, like the other subprograms of the algorithm, is developed under the guideline of retaining variable values instead of constant values in the computer code wherever possible. Expansion to other contingency planning problems is provided for by these variables except for changes to the size of the file buffers and the array dimensions which are normally at a fixed level for a specific run.

The contingency model's goal equations consist of eight distinct constraint sets as described in the third chapter. It is natural to generate coefficients of the goal equations in COEF by splitting the total number into eight segments and developing one constraint set at a time. The number of rows of equations generated for each set varies; one constraint set produces as many equations as there are job categories, a number which we designate as JOBCAT; two sets produce as many as there are time periods, a number which we designate as JTIMES; the remaining sets require an equation for each job category and for each time period and, thus produce as many as there are in the product of JOBCAT and JTIMES, which we designate as NEW.

The initial TE matrix is established with the decision variables heading the lefthand-side columns as nonbasic variables and the positive deviation variables heading the righthand-side columns as nonbasic variables. (A variable heading a column means that the coefficients in successive rows of the column belong to the variable in the goal equations.) The way that the decision variables are ordered across the top of TE is arbitrary; our setup, upon which COEF is oriented, is the following block structure, left to right: x's, c's, v's, y's, u's, w's, and e's, where these symbols are defined in the third chapter. Each block of variables, except for the e's consists of a variable for each job category and each time period and is arranged thus: the variables of the first time period from the first

job skill to the last job skill, the variables of the second time period from the first job skill to the last, ..., the variables of the last time period from the first job skill to the last. (This arrangement is left to right in the block.) The block of e's consists of an e variable for each time period arranged from left to right from the first to the last time period; no job skill shredout is applied to these variables, the flows from the national labor supply into basic training. The positive deviation variables are ordered left to right in succession from the variable of the first goal equation to the variable of the last goal equation.

Zeros dominate the initial technology matrix. The righthand-side submatrix of positive deviation variables is identified as a negative identity matrix of order equal to the number of goal equations. The lefthand-side submatrix is neither upper nor lower triangular; however, a pattern is evident. The repeated use of coefficients of early time periods in a given constraint set appear to expand more nonzero coefficients into rightward and higher-row-number positions of the TE block for that set. In these constraint sets that produce NEW equations, the setup in TE conforms to rows of equations for time 1, time 2, ..., time JTIMES and within each of these smaller sets for a time period from job category 1 to job category JOBCAT. With this setup, each column block of variables included in a constraint set shows this pattern to be lower

triangular. (See Figure 4-3.) Each new subdiagonal below the block's main diagonal is positioned a total of JOBCAT units away from the next diagonal; also, a total of JTIMES diagonals (including the main diagonal and every subdiagonal) exists for each block of decision variables pertinent to the constraint set.

COEF is designed to produce this lower triangular pattern for the constraint sets of NEW equations by two DO loops, one nested inside the other with the same terminal statement for both loops. The outer loop of the form DO sn J=1, NEW iterates on one column of the block of variables at a time; J increments by one only after all coefficients of the constraint set are in position in that column. (sn in the DO loop is the terminal statement of the loop.) The inner loop of the form DO sn I = J, NEW, JOBCAT iterates on a particular starting row and increments in steps of JOBCAT, reflecting the relative position of the subdiagonals. The lower triangular pattern is guaranteed by the fact that the inner loop never iterates on a value less than the column parameters. It is essential to realize that J refers to horizontal movement in the block and I to the vertical movement in the block. Increasing J implies dealing with variables of differing job categories and increasing time periods; increasing I implies dealing with goal equations of later time periods. For a further discussion of this nested structure, we

advise the reader to examine the comments of COEF of the computer code located in Appendix B.

COEF has the capability to modify any negative righthand-side values with the variable SWITCH. Certain policy configurations may generate negative righthand-side goal equations values in the second constraint set. As Ignizio warns, "However, the modified simplex algorithm requires that all righthand-side values be nonnegative [Ref 15:57].". If the analyst or decision maker recognizes that this limitation has been observed with the policy that he has formulated, he sets a nonzero value for the input variable SWITCH. A nonzero SWITCH forces the algorithm to multiply the righthand-side values of the goal equation and the lefthand-side TE values by -1. A zero SWITCH bypasses this multiplication. Where negative righthand-side values develop, care is necessary in the interpretation of the negative and positive deviation variables; the original positive deviations of the second goal set are initially in the basis.

The subprograms of the updated model conform to certain standards. All subroutine calls are calls by reference in which the address of the argument is passed to the subroutine through the argument list of the CALL statement. No matrix is an argument of any subprogram. The essential matrices for the algorithm lie in blank COMMON or labeled COMMON blocks which provide the majority of communications between different program units.

The design of this source code is also an effort in efficiency with features for saving time in central processing as well as in input/output processing and for conserving the use of central memory. Problems in running the program develop as the model expands to accommodate more time periods and job categories. More memory is needed; hence, ways to reduce memory requirements are employed. The buffer size of every file, inserted in the initial PROGRAM statement, is normally set at the default value of $2002_8 = 1026$ words in the CDC 6600 computer. This quantity is ample for normal input/output processing, and optimization by the FORTRAN compiler helps to determine an optimal size as well. A smaller buffer size, however, decreases overall memory allocation but at the expense of increased input/output processing time. The buffer size is strictly dependent on the size of the particular contingency problem to be solved and on the number of records of information processed at any instant of time during execution. A trade-off in the conservation of computer resources must be resolved, because as the buffer size decreases central memory allocated to the run, the input/output time tends to increase. The converse is likewise true. Considering our file buffers, we find that a smaller input file buffer and output file buffer of 300_8 words (192 words) for small contingency problems (such as a test case of two skills, three time periods) save space with little increase in input/output processing time.

The use of ENTRY statements (nonstandard FORTRAN IV features available with the CDC 6600 machine) helps to reduce computer memory. As subroutines TEST and PERM have identical subroutine arguments, a single subroutine with two entry points satisfactorily performs the same work as the individual routines. This structure results in using 18 fewer computer words with both parts of the routine having access to the same COMMON block which is declared only once. Similarly, subroutine ERRORS is an alternate entry point for the program unit with entry point POUT; both routines require one argument in the CALL statement.

Another consideration to reducing the number of computer words is word packing. This technique aims to use more available space in a computer word by the storage of more than one value. The word-packing and bit-manipulation capabilities with the CDC 6600 machine deal only with integer variables. The structure of the computer word used for integer values is the complete 60-bit configuration without an exponent. For example, in array of positive integer values not greater than $512 = 2^9$, each value requires no more than 10 bits, and six values can be stored in one 60-bit word. The array space required for this simple example is reduced to one-sixth of the usual unpacked array space. Unfortunately, the simplex algorithm does not benefit from this technique, for most arrays of the algorithm are real-valued. The four integer variables in blank COMMON seldom change, and although they

may be packed into one word, the savings of three words of storage is hardly substantial.

Masking expressions with the use of the SHIFT and MASK functions (both nonstandard FORTRAN features) are beneficial for reducing execution time when repetitious multiplication and division by powers of 2 or the repeated use of the modulus function of the form $\text{MOD}(n, 2^k)$ occurs. The potential for saving memory by this method is not evident.

In a test of the modified algorithm on a small model, the following statistics were generated:

36576 ₈	words alone to execute the program
40500 ₈	words to load and execute the program
14.992	seconds central processing time, including 6.487 seconds of time to compile the source code
22.921	seconds input/output processing time

The final solutions match those provided by the original algorithm developed by Ignizio. Coefficients in the original model are read into TE (hence, influencing input/output time) and are calculated in the updated version (hence, influencing central processing time). In the updated version, MAIN, INIT, and COEF accomplish what the original GMAIN used to perform, yet the number of words of code generated during compilation for the three subprograms is greater than that for GMAIN. The two models differ noticeably in design.

In order to reduce computer memory further, we use segmentation loading. In some computer codes, the program units need not reside in memory simultaneously. Two distinct phases exist in our source code. When initialization occurs, testing for entering and departing basic variables is not necessary. Initialization routines are unnecessary once the execution phase begins operations. Segmentation is an efficient way of loading compiled subprogram modules into central memory so that compiled program units which are not involved with the current processing are not loaded into memory; hence, those units uninvolved are not occupying extra space in the machine. The binary subprograms are arranged by segmentation directive statements into "tree structures." These subtrees share memory.

Segmentation is capable of significantly reducing the amount of central memory needed by large programs. It is much more powerful, efficient, and flexible than overlay for large core programs and is therefore recommended for this class of problem [Ref 2:5-13].

Communication among the segments is two-way unlike the one-way communication of overlay loading. Labelled COMMON blocks are considered as well by the segmentation directives GLOBAL. The segments include a "root segment" which contains the MAIN program. Out of this root segment are branches containing other modules of the particular program. The segmentation load for the updated model is depicted in Figure 4-4. The directives that create this structure are also presented.

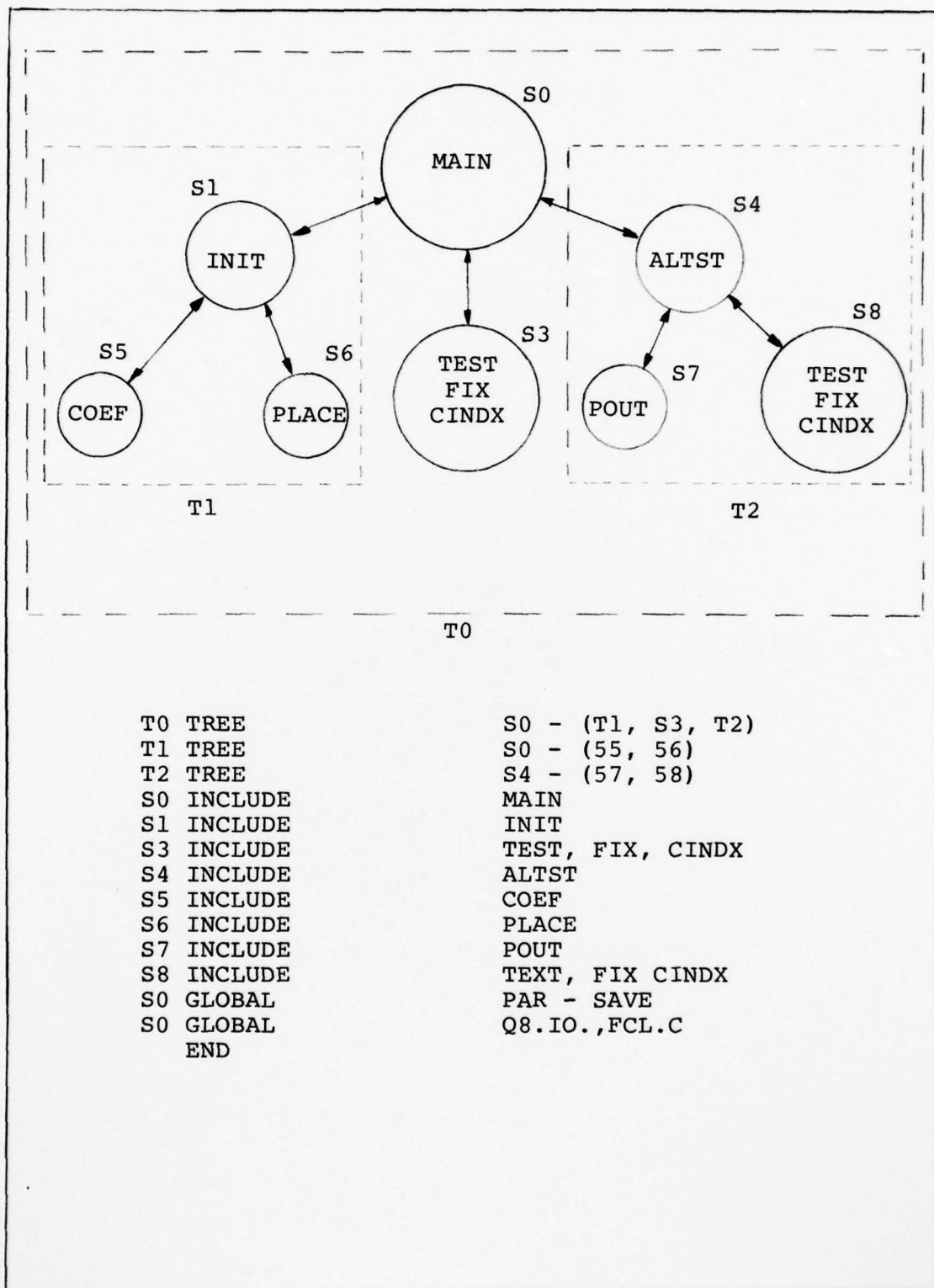


Figure 4-4. Segmentation: Tree Structure and Directives

For the updated design, the segmentation structure includes two subtrees and a branch stemming from the root segment. One subtree T1 deals with the initialization. During this phase, COEF is called, provides TE coefficients and resumes control to INIT. INIT directs the reading and placement of objective function values with PLACE. The split between COEF and PLACE is inherent; hence, the tree structure involves two branches out of the segment S1 where INIT resides.

The execution phase is divided into two logical parts: (1) a testing/permuting phase, and (2) an output phase. This division appears in the structure by segments S3 and T2.

It is important to understand that each branch of the structure (S0-S1-S5, S0-S1-S6, S0-S3, S0-S4-S7, or S0-S4-S8) is mutually exclusive from another branch. Thus, only one branch occupies central memory at one time. The blank COMMON block which is first declared in MAIN is accessible to all segments for it resides in the root segment. Communication is preserved as it is in the normal loading scheme.

The same small case above is now run with a segmentation load and reveals improvement in the use of core memory; however, the disadvantage in this run lies with increased input/output processing time.

SAVINGS

35751 ₈ words alone to execute	625 ₈ words
34300 ₈ words to load and execute	4200 ₈ words
15.346 seconds central processing time including 5.859 seconds to compile the source	- .354 sec.
30.313 seconds input/output process- ing time	-7.392 sec.

This run requires a minimum of 35751₈ words, while the normal load needs 40500₈ words, a total savings of 2527₈ = 1367 words of central memory. Whether or not an analyst employs segmentation to the updated model is a choice involving the tradeoff between decreased memory versus increased input/output time. As the problem size expands, the tradeoff widens.

Use of the Simplex Algorithm on the Case of Three Skills and Six Time Periods

The computer algorithm is developed to the point where we can solve the first numerical example outlined in Chapter Three. Initial conditions specified in that chapter are read into the program and the algorithm is executed in normal and segmented loading schemes. Unit weights are assigned to every element of the objective function.

The results indicate that all goals have been satisfied except for the fourth priority level goals ($a_4 = 171.758$) and sixth priority level goals ($a_6 = 11.290$).

We have not fully met our CONUS requirements during some of the time periods and simultaneously have exceeded CONUS requirements during other time periods. All overseas theater requirements have been fully achieved. Table IV-1 summarizes our resulting CONUS and overseas levels.

CONUS requirements are exceeded slightly during the first and second time periods for skill types 1 and 2. Our CONUS levels drop below CONUS requirements during time period 4 for skill type 1 and during time periods 3-6 for skill type 2. CONUS requirements are fully satisfied for skill type 3. At time $t = 6$ only fifty-three and six-tenths percent (53.6%) of skill type 2 CONUS requirements are met.

Obviously, our solution has resulted in a poor achievement of skill type 2 (direct mission support) CONUS goals. This may seem unusual because we assigned no extra weights in the objective function in favor of the other two skill levels. However, our problem structure is implicitly weighted away from the second skill type for three reasons. First, we desire that skill type 2 levels increase by a much larger amount overseas than the amounts required for the other skills. Second, the length of technical training for the second skill type is twice as long as the length of technical training for the other skills, creating a longer delay in satisfying skill type 2 goals. This delay has direct influence on goal quotas to be met later in the contingency. Third, while the desired rate of increase

Table IV-1

Resulting CONUS and Overseas Levels
For First Numerical Example (in Thousands)

t	i	Scenario Requirements		Solution Levels	
		CONUS	O/S	CONUS	O/S
1	1	75	29	75	29
	2	152	42	157.8	42
	3	170	32	170	32
2	1	75	33	78.8	33
	2	152	60	153.7	60
	3	170	35	170	35
3	1	75	50	75	50
	2	152	83	143.9	83
	3	170	40	170	40
4	1	75	70	58.8	70
	2	152	113	122.1	113
	3	170	45	170	45
5	1	75	85	75	85
	2	152	138	104.9	138
	3	170	60	170	60
6	1	75	95	75	95
	2	152	165	81.4	165
	3	170	75	170	75

t represents the time period.

i represents the skill type.

for skill type 2 overseas is much higher than for the other skills, the allocation of reserves among the three skills does not reflect this disproportionately. Reserves are allocated equally among the three skills.

Table IV-2 provides the various flows and instructor levels resulting from the simplex solution. Flows into technical training $w_{i,t}$ during the time interval $(0,3)$ reflect the peacetime and pre-conscription training pipeline. Significant flows into skill type 2 technical training occur initially because of rapid skill type 2 growth requirements. Also, at least during the initial time periods, CONUS requirements for other skills are met by the reserves and the initial training pipeline. However, during later time periods ($t \geq 4$) the growth requirements for skills 1 and 3 also become more significant. Because their technical training times are shorter, skills 1 and 3 begin receiving the majority of trainees. The shorter training times enable quicker goal satisfaction. Moreover, beginning at $t = 5$, gains into skill type 2 technical training have no influence on meeting CONUS and overseas requirements.

It is also noteworthy that the third skill type instructor pool appears to be a sink for absorbing personnel who are excess to CONUS needs. The excesses are personnel who are conscripted after CONUS requirements are met to the maximum extent. Recall that we do impose the requirement that all available labor be utilized during

Table IV-2
Resulting Flows and Instructor Levels for First
Numerical Example (in Thousands)

Time Period	Skill Type	$x_{i,t}$	$c_{i,t}$	$y_{i,t}$	$u_{i,t}$	$v_{i,t}$	$w_{i,t}$	e_t	$I_{i,t}$
1	1	4.75	0.	7.591	.105	4.615	1.672	11.67	4.719
	2	4.76	0.	10.	.317	2.967	3.17		3.284
	3	2.60	0.	0.	0.	.73	3.327		1.063
2	1	4.87	0.	10.	2.36	5.364	0.	50.	7.723
	2	18.84	0.	10.	1.642	0.	8.169		1.642
	3	3.64	0.	0.	.532	.219	0.		.7495
3	1	17.66	0.	10.	3.862	0.	0.	50.	3.863
	2	23.78	0.	10.	.821	0.	8.169		.821
	3	5.35	0.	9.73	.375	4.755	0.		5.1295
4	1	20.5	0.	2.409	1.931	0.	30.894	50.	1.932
	2	30.415	0.	0.	.411	0.	4.106		.41
	3	5.04	0.	10.	2.565	7.525	0.		10.09
5	1	15.7	0.	0.	.965	0.	10.367	50.	.967
	2	25.565	0.	0.	.205	0.	0.		.205
	3	15.045	0.	10.	5.045	0.	24.633		5.045
6	1	10.85	0.	0.	.483	0.	7.724	50.	.484
	2	27.69	0.	0.	.103	0.	0.		.102
	3	15.06	0.	0.	2.52	12.095	27.276		14.62

the last two time periods. Most gains into technical training during the last two time periods are in skill type 3 because more skill type 3 instructors may be obtained from the CONUS. While we desired that CONUS requirements not be exceeded, we placed no upper limit on the number of instructors.

The solution discussed is one of several (25) alternate solutions resulting from the problem. Each optimal solution is basically the same result as presented above with only minor differences in certain decision variables. Thus, for each alternate solution, the objective function satisfaction levels are identical, and the resulting CONUS and overseas levels are identical. The difference between successive solutions occurred with the direct flows of personnel from the CONUS force to the instructor pool, from this pool to the CONUS force, and the activation of reserve forces for use in the CONUS force. In all cases the differences between solutions were compensating and yielded basically the same result. For instance, one solution differed from the first printed solution in the variables of the second time period. During that time period, 1200 fewer persons left the CONUS to become instructors. Simultaneously, 2400 fewer persons in the reserves were activated coupled with 1200 more instructors leaving their technical training positions for assignments to the CONUS. The CONUS force, hence, remained unchanged from the original strength allocation of the first optimal solution;

the composition of that force was created differently by each successive solution. The fact that no upper limit was placed on the instructor pool contributed to the generation of alternate, but basically the same, solutions.

The segmentation run showed identical results as anticipated. The comparison of computer source utilization follows in Table IV-3. The creation of the segments for segmentation loading contributes to the higher input/output time. Savings in memory over increasing time is an evident tradeoff.

Table IV-3
Comparison of the Two Loads

	Normal Load	Segmentation Load
Memory (words)		
to execute	107050 ₈	105654 ₈
to load and execute	35700 ₈	31100 ₈
Processing Time (sec.)		
central	522.355	524.791
input/output	23.385	50.233

Now we attempt to improve the achievement of skill type 2 CONUS requirements by utilizing appropriate weights in the objective function. Specifically, we assign weights of 10 to skill type 2 CONUS negative deviation variables for $3 \leq t \leq 6$. Additionally, we assign weights of 5

AD-A065 909

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/6 12/2
PERSONNEL CONTINGENCY PLANNING MODEL USING GOAL PROGRAMMING.(U)
DEC 78 J A MORENO, B W UTZ
AFIT/60R/SM/78D-10

UNCLASSIFIED

NL

2 OF 4

AD
A065909



to skill type 1 CONUS negative deviations for the same time periods. We retain unit weights for skill type 3.

Table IV-4 provides a comparison of the solution using these different weights with the first solution. In both cases, the overseas goal requirements were fully met.

Our second solution provides sixty-two and four-tenths percent (62.4%) achievement (as compared to fifty-three and six-tenths percent (53.6%) for the first solution). CONUS requirements are underachieved now at $t = 4$ and $t = 5$ for skill type 1 (as compared with underachievement for this skill type only at $t = 4$ in the initial solution). Even with unit weights assigned to skill type 3 deviation variables, implicit weighting is still in favor of the third skill type. This circumstance is probably due to the fact that the attrition rate, the instructor needs, and the buildup requirements are at the minimum for the third skill type. Effective measures for reducing the number of redundant alternate solutions are provided later in the thesis.

Further Algorithm Development: Employment of Core-Swapping

As we attempt to treat larger problems and expand the contingency to a situation involving three job categories over twenty time periods, difficulties in size of storage capacity result. For the earlier cases, adaptation of array dimensions and file buffer sizes pose no problems. The blank COMMON block, though, for the new

Table IV-4

Comparison of Solutions: One Without Weights and
One With Different Weights Among Skill Types

t	i	CONUS	Levels	
		Requirements	Test 1	Test 2
1	1	75	75	75
	2	152	157.8	154.3
	3	170	170	170
2	1	75	78.8	76.6
	2	152	153.7	152
	3	170	170	170
3	1	75	75	75
	2	152	143.9	141.2
	3	170	170	170
4	1	75	58.8	57.7
	2	152	122.1	120.7
	3	170	170	173.2
5	1	75	75	60.8
	2	152	104.9	104.2
	3	170	170	170
6	1	75	75	75
	2	152	81.4	94.8
	3	170	170	170

t represents the time period.

i represents the skill type.

Test 1 had no weights applied.

situation enlarges significantly. For the smallest test case, the COMMON block is a storage area of 12706 words; overall field length to load and execute is 16704 words which is reasonable. For the new situation, the COMMON block resembles the following and requires 261968 words:

```
COMMON  TL(343,6), TT(6,723), TE(343,723), TI(6,723),  
        TB(343), TA(6), JCOL(723,2), JROW(343,2),  
        X1(380,2), NOBJ, NPRI, NVAR, NCOL
```

The total capacity of the CDC 6600 computer is 131072 words. It is impossible to continue using the current updated model with this excessive demand for storage.

The majority of the space in blank COMMON is occupied by the TE matrix (247989 words). This matrix is sparse but the position of coefficients is crucial for the pivoting routines of the method. Knowing which variables leave and enter the basis during a given iteration at a priority level must be retained. Dealing with the formidable size of TE requires the capability to separate TE into smaller blocks and then use each block where necessary. The program must adapt the capability of reading and writing new values of TE after each call to the permuting subprogram. Now the question we impose is: How should the TE matrix be stored and retrieved?

Three forms of available input/output processing are possible answers: (1) mass storage, (2) buffer input/output, and (3) unformatted input/output. Mass storage is beneficial whenever a program requires that information

be accessed in a random fashion. This process tends to be expensive and prohibitive, for the mass storage sub-routines require excessive input/output time. As we observe the input/output requirements of the subprograms that access TE, the data is needed in a sequential manner (from the first row of TE to the final row of TE), and mass storage becomes a less attractive option for our solution.

The second alternative saves central memory allocation by eliminating the space required of the file buffer. The comparison of input/output processing time of this approach with that of unformatted input/output leads to the abandonment of buffer input/output for the solution of the problem. The choice of unformatted input/output allows us to specify variable lengths of computer records to be processed as well. Because each constraint set has one of three possible lengths of rows (JOBCAT, JTIMES, or their product, NEW), unformatted processing permits variable length records and can process a specific constraint set more easily than buffer input/output. Not every element of the TE array in blank COMMON has to be accessed with the use of unformatted processing whereas buffer input/output requires every element of TE to be processed at once. Storage of information is larger when using buffer input/output statements than when using unformatted files.

The updated version of separate phases is modified to accept core-swapping with unformatted input/output. (Core-swapping is the technique of keeping an entire

portion of a matrix on file but not in central memory at once; parts of the matrix are processed one at a time so that the memory space allocated to the array is small.) The single main program of the updated model is divided into three main programs. The first main program includes the initialization phase without the establishment of the arrays of initial objective function values. These arrays are excluded from the blank COMMON block and reduce the allocation of this program. Only TE and the array of right-hand-side values exist in this program.

As the largest number of rows in a constraint set which are generated at a time by COEF is NEW, the space for TE in COMMON must have that number as the row dimension in the COMMON statement. The number of necessary columns of TE may be reduced further to be either the number of decision variables (NVAR) or the number of goal equations (NOBJ), whichever is larger. Thus, TE is split up and shortened into blocks as large as JOBCAT*JTIMES rows and max(NVAR,NOBJ) columns, either the lefthand-side of TE or the righthand-side of TE. (See Figure 4-5.) For the large case under consideration, the TE block assignment in COMMON is 60 rows by 380 columns for a total of 22800 words.

An additional feature allows for more than these 16 blocks of TE to be generated during this initialization program. The integer variable INDEX is read in to allow the program to divide the generated block of a constraint

Initial Decision Variables	Initial Positive Deviation Variables	
		Constraint Set I
		Constraint Set II
		.
		.
		.
		Constraint Set VIII
NVAR	NOBJ	

Figure 4-5. Breakdown of Split Technology Matrix

set into smaller blocks of a maximum of INDEX rows each. For instance, if INDEX = 20 and we deal with this "3 by 20" case, the first constraint set generates a block of 60 equations, but the program divides this block further into three blocks of 20 rows each prior to writing the block to the unformatted file. Other constraint sets follow similarly, and for instances like the third constraint set where only three equations are created, the block generated in COEF is written to the file as it is. The program tries to split a generated block into as many sub-blocks as possible with an INDEX number of rows. Hence, the execution phase of the algorithm (the second and third main programs) requires a TE matrix allocated in blank COMMON equal to INDEX rows and max(NVAR,NOBJ) columns.

Three new subroutines--BEGIN, FINISH, and RHSTE--are constructed to generate the blocks on the unformatted file. As this phase operates, the entire lefthand-side of TE is written on the file. Each generated subblock is a record of the file. After the eighth constraint set is complete, the righthand-side of the TE matrix is similarly divided into associated blocks like those of the lefthand-side and written on the file. Once the TE matrix is processed, the array of righthand goal values becomes the final record on the file.

Deciding what value of INDEX to select takes experimentation. There is no fixed rule for the decision. We must remember that the smaller the INDEX, the smaller the TE dimensioning and the more blocks processed on the file. More blocks processed on the file implies greater input/output processing time in the execution phase. The larger the INDEX becomes, the greater amount of storage in blank COMMON is required but at a decrease in time for the input/output functions.

For the "3 by 20" case with an INDEX of 20, 36 blocks of initial TE coefficients are created on the file. The matrix of righthand-side goal equations coefficients which are calculated during initialization is written as the final block of data on the unformatted file. The first main program then requires 12.909 seconds execution time and 31.660 seconds input/output time. The field length necessary to load and execute the program is 112731 words.

Concerning the retrieval of TE values from the file and using the values correctly with other matrix values, the model needs a method to calculate the correct subscripts of the original TE matrix and the matrices always resident in core. Subscripting the variables grows more complex since core-swapping is the way of handling the storage capacity problem. (Core-swapping is the technique of processing information in a large array without having the entire array in memory at one time. This array is kept complete on an external file on magnetic disk or tape, and the records of the file which are subblocks of the array are read and processed one at a time.) Once the program determines which TE coefficient is requested, it must know whether or not that the current TE block in central memory contains the requested value. If the current block is not the "right" one, that block which is "right" must be read into central memory, and the coefficient can be retrieved directly from core.

Concerning the updating of the TE matrix on file, the program generates new values each time that PERM is called. Once PERM ends, the file must be correct so that the program can continue searching for the optimal solution. It is impractical, though, to construct a function subprogram to handle this retrieval and updating process for only one value at a time. Instead, whole blocks are updated at a time. The elements of the TE matrix are not randomly chosen; subroutine CINDX requires every value in row after

row of TE in order to compute new index rows; subroutine PERM pivots and simultaneously updates each TE element in a row-by-row fashion. Instead of relying on a function subprogram to retrieve TE values singly, we must develop the structure in these extant subprograms to read each record of TE, modify or access it, and continue sequentially until the final record of TE is processed. The structure includes reformulating double summations, especially where the inner summation iterates over TE row subscripts. (Sometimes this untangling of indexes of double summations is impossible to reverse, but the process is successful with CINDX, PERM, TEST, and ALTST.)

To accomplish this development, a new array referred to as IN is introduced into COMMON to keep a historical set of information on the initial creation process of the unformatted file. IN is dimensioned eight rows by three columns, where each of these eight rows refers to a particular constraint set of TE. The first column indicates the original number of equations generated in the constraint set, the second indicates the number of rows contained in the final subblock of the set, and the third indicates the number of subblocks created for the set. The computer code is adapted through reordering of the DO loops to handle both the lefthand-side of the TE matrix (first half of the file) and the righthand-side of the matrix (second half of the file) in that order. Each block is read in and processed. Correct limits on DO loops for reading

the unformatted file are specified by integer variables and change only when the final block of a constraint set is about to be read.

The program is modified to handle the updating of the TE matrix in PERM by reading from one unformatted file to a second one. The integer variable NTAPE introduced in blank COMMON refers to the file ready for reading and the local integer variable NSCRAT refers to the file ready for writing the corrected TE records. Once PERM is near the conclusion of permutation, the variable NTAPE is set to equal NSCRAT; in this manner, the other subprograms use the most up-to-date information of the TE coefficients. Input/output time is saved by not having to rewind the original file and rewrite the new information onto the same file.

To maintain the pivoting structure as before, communications between PERM and the subroutines TEST and ALTST require a vector of the coefficients of the TE column of the entering basic variable. Both TEST and ALTST create this vector prior to their subsequent calls to PERM. (ALTST must create this vector twice because of the second call to PERM in its program logic.) PERM uses this vector and creates a second vector of coefficients of the row of the departing basic variable. PERM requires both vectors of coefficients in permuting the TE matrix and prevents the unwanted alteration of these original values before the permutation process is complete. The original values are

necessary to achieve permutation. The problem does not exist in the updated model where the entire technology matrix resided in central memory.

Now that the storage capacity of TE is reduced, it is possible to reduce the required central memory further. The index rows are always created by a call to CINDX. Once the index rows are in memory, the top stub values which produce this set of index rows are not needed until the flow of control passes to PERM. We observe that the top stub values and the index rows have identical dimensions in COMMON and are mutually exclusive matrices; when one is required in memory by a subprogram, the other is not referenced. Thus, we can eliminate the extra space declared for one of the matrices, allow both sets of values to occupy the same locations in core when needed, and refer to the mutual array as TT (formerly the reference to top stub values).

After initialization has positioned the objective function weights in TT, it is necessary to copy these values to a file. The choice of buffer input/output on this file saves extra memory by eliminating the file buffer. After the copy is complete, the integer variable NTT signifies which set of values is in memory at the present time. This switch indicates whether or not top stubs are in memory, for only these values are read in or written on the file. In subprogram CINDX the index row values evolve through computation with the current top stub values. The switch

is set accordingly. In subroutine PERM, the top stub values are changed because of the departure of and entrance of variables in the TE matrix into the basis. The file of current top stub values is rewound and read into memory. While in central memory, the changes are enacted, and the new information is buffered out to the file, ready for a subsequent call to CINDX.

The second main program of the new algorithm begins with an initialization routine to process the objective function values and to read in variable labels. Because central processing time increases as a result of the processing of the large technology matrix, the main program uses the SECOND function to determine if a length of processing time has elapsed. Once this length of time is reached, control flows to a newly designed subroutine STORE which stores the current TE values on one file and all remaining COMMON information along with the current priority level on a second file.

The third main program of the new algorithm differs from the second main program in a few respects. Instead of an initialization routine to read initial values into matrices, this third program begins by restoring the arrays and integer variables in COMMON to the values that they assumed during the run of the second main or in a previous run of this program. Thus, subroutine PLACE is deleted from the third main program. Subroutine RESTOR is an alternate entry point for subroutine STORE. When the

computations in MAIN dealing with elapsed time indicate that a predetermined quantity (3000 sec) has been reached, the third main program calls subroutine STORE to prepare the files with the latest values. Repetition of this program is necessary until no new optimal entering variable results for the last priority level.

The final design of the core-swapping algorithm is depicted in Figure 4-6.

Running the Final Model

In order to execute this "3 by 20" contingency model on the CDC 6600 computer, we are led to use magnetic tape as a medium of storage. Two tapes are necessary for holding the TE matrix and the COMMON block data.

During the testing of the model, we encountered several hardships in dealing with the computer--other users' priority runs, computer downtime, and unforeseen coding errors. One significant problem is the storage of our initial and subsequent TE matrices on magnetic disk. The TE matrix for this model requires nearly 490 blocks of computer allocation. Trying to make this data file permanent for later runs is an impossible task when we consider magnetic disk space. The data file of TE elements is large enough to begin dumping earlier permanent file information. The problem does not materialize, however, with the use of magnetic tape.

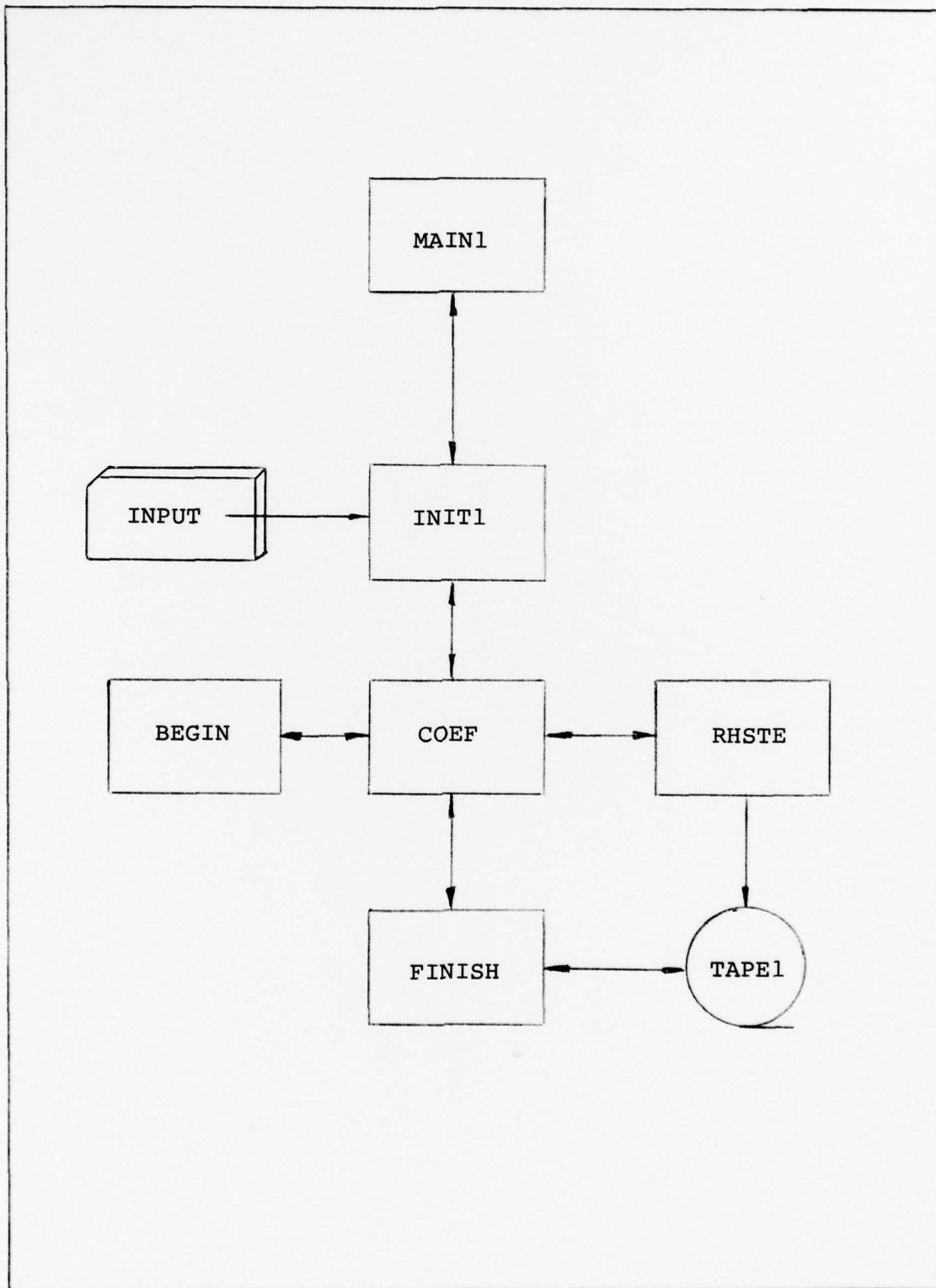


Figure 4-6. Final Model for Large-Scale Contingencies

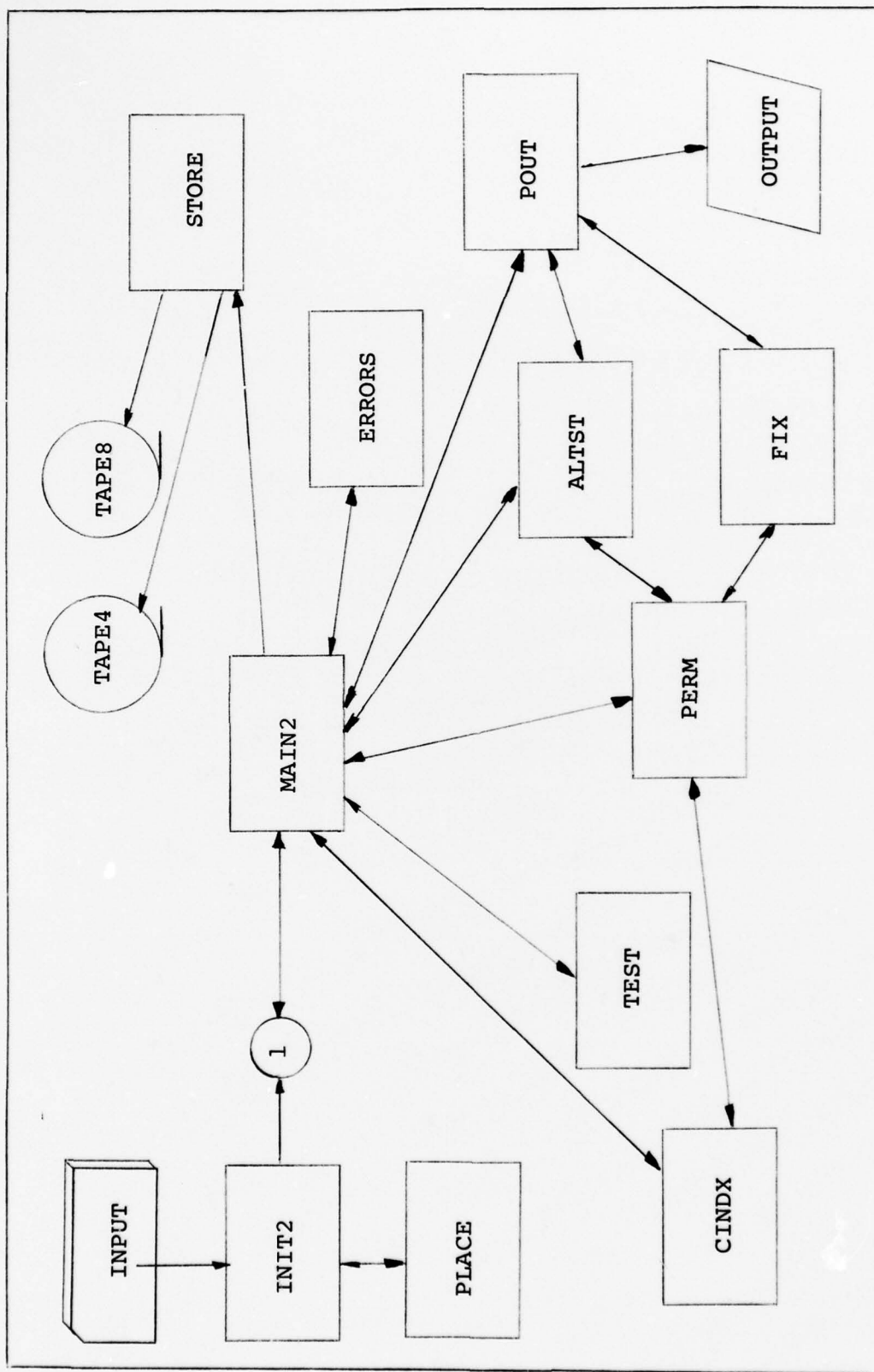


Figure 4-6--Continued

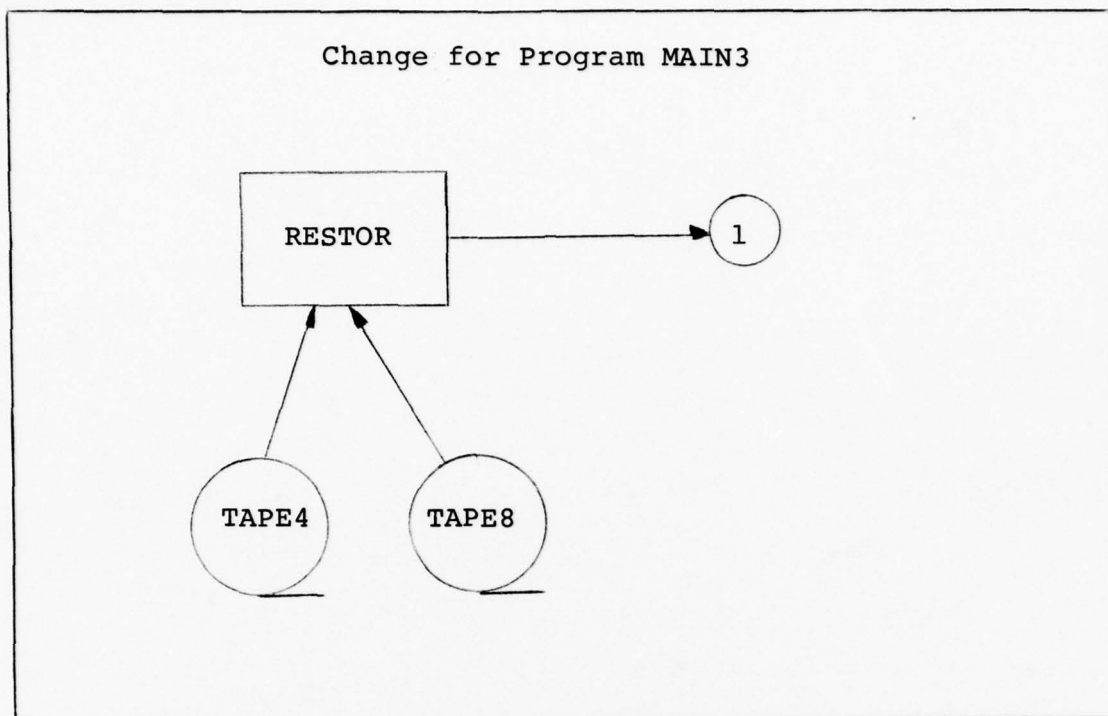


Figure 4-6--Continued

With the first main program, one tape is necessary for storing the initial TE coefficients and righthand-side goal values. This local file is referred to as TAPE1. During the second main program the tape file is called TAPE1 again, and by the time that STORE prepares to retain the most current TE data, the tape file is mounted on the tape drive during this program; by the time subroutine STORE ends, the remaining COMMON block data is written on the second tape, and the file is referred to as TAPE4.

The third main program requires both tape files to restore the data and begin anew. During this program's execution, TAPE8 is written to TAPE1, a magnetic disk file which is not a permanent file. TAPE1 and TAPE2 are the

two local files which alternate in subroutine PERM as the tape with the most current TE coefficients.

Similarly TAPE4 initializes the magnetic disk file TAPE3 with TT values. TAPE3 and TAPE4 are processed by buffer input/output commands. Keeping TAPE3 as well as TAPE1 and TAPE2 as local files on magnetic disk saves input/output processing time from having to rewind magnetic tapes and write on the slower medium of magnetic tape.

Fortunately, the modified simplex algorithm allows the user to obtain a global optimum solution according to the implicit policy structure. Infeasible intermediate basic solutions are cause for program termination, a safeguard in the algorithm. Running the case of three job skills over twenty time periods is a slow process despite the conservation features implemented in the overhaul of the updated model. The effects of enormous storage capacity are felt in the other computer resources of time.

A numerical test example is run with an objective function similar to that in the earlier example of 3-skills, 6-time periods. Now, policy objectives include a buildup of the force to double the theater force size over the last eight months of the conflict. With an INDEX value set at 20 (allowing no more than twenty rows of TE matrix to be in central memory at one time), the three main programs require a maximum field length of 164046₈ words for loading and execution. The savings in computer memory is offset, though, by several hours of central processing

time. Five hundred total iterations are necessary to test and permute the TE matrix for the six priority levels assigned. Because of the sequential access of the data on file, the permutation routine handles a quarter of a million calculations on every iteration. The routine CINDX which computes index rows needs to access the file for every iteration from the first priority level to the current priority level. The iterations at the sixth priority level take longer to accomplish because of the necessity to access the data more frequently (from the first to the sixth priority level).

The final solution seems cogent with roundoff error kept at a minimum. Observing intermediate solutions failed to detect roundoff error beyond the eighth decimal place (a value equivalent to .00001 man). The final solution, though, shows some roundoff error in the fourth and fifth decimal places but not an alarming result.

Because our programming with simplex consumes more computer time than permissible for accurate and adequate analysis, we decided to try an alternate approach, the pattern search techniques for goal programming. These techniques become the focus of our next chapter.

V. Development and Testing of the Pattern Search Algorithm

Introduction

In this chapter we develop and use an alternate solution procedure for our model which is derived from James P. Ignizio's pattern search method for nonlinear goal programming (Ref 15:166-175). Ignizio's method is based on an extension of the direct search method of Hooke and Jeeves (Ref 14:202-229) and is a fairly easy method to employ or to program. The method is selected primarily because it requires a great deal less computer memory and computation time than does the simplex procedure for large problems.

The Pattern Search Procedure

Ignizio's pattern search procedure consists of three major components: the Hooke and Jeeves search algorithm, the system of goal equations with righthand-side values, and the objective function. We established the general expressions for the last two components earlier in this research. Prior to initiation of the search algorithm, two more items are needed. We need an initial guess for each decision variable so that we may establish a base point from which to begin our search. We also need a step size for each decision variable to identify how each

variable is incremented during the search procedure. Normally, the better our choice of an initial base point, the faster the search will converge to an optimal solution.

The Hooke and Jeeves search algorithm begins at the initial base point. For our particular model the base point \bar{x}_{base} consists of an initial guess for each decision variable ($x_{i,t}$, $c_{i,t}$, $v_{i,t}$, $y_{i,t}$, $u_{i,t}$, $w_{i,t}$, e_t , $v_{i,t}$). Each priority level component of the objective function is evaluated at the base point to provide the initial satisfaction of goals at each priority level. This action is accomplished by the following procedures.

1. The goal equations are evaluated in terms of the initial guess for the decision variables.

2. The values of the goal equations are compared with the righthand-side values, and the deviation variables are evaluated.

3. Each priority level component of the objective function is evaluated in terms of the applicable deviation variables multiplied by the assigned weights.

The initial values of the various priority level components of the objective function provide the initial base criterion for determining whether or not improvements are made in the objective function during the search.

Having established the initial base point and initial evaluation of the achievement function, we now begin the search procedure. The initial decision variable (in our model, $x_{1,1}$) is increased by an amount equal to the

variable's designated step size ($x_{1,1} = x_{1,1} + \delta_{x_{1,1}}$). After this first step we reevaluate our goal equations and the objective function to determine if an improvement in the objective function has occurred. Examples of improvement include:

1. An improvement in the first priority level goal achievement.
2. No change in the achievement of first priority level goals, but an improvement in the achievement of second priority level goals.

Examples of no improvement include:

1. No change in achievement of any priority level goals.
2. Improvement in the achievement of second priority level goals, but a worsening in the achievement of first priority level goals.

Moreover, our preemptive priority goal structure is in effect. We attempt to satisfy higher priority level goals prior to considering lower priority level goals. We attempt to improve lower priority level goal achievement only if we do not worsen higher priority level goal achievement.

If an improvement in the objective function has resulted from our step increase in $x_{1,1}$, then $x_{1,1}$ will retain its new increased value. If no improvement has resulted, we will then decrease the original value of $x_{1,1}$ by an amount equal to the step size specified for $x_{1,1}$

$(x_{1,1} = x_{1,1} - \delta_{x_{1,1}})$, reevaluate our goal equations and the objective function, and determine if an improvement has resulted from this action. If there has been an improvement over our initial evaluation of the objective function (based on our initial base point), then $x_{1,1}$ will retain its new decreased value. If no improvement has resulted, $x_{1,1}$ will be returned to its original base value. The value of the objective function against which further comparisons of attempts for improvement will be made is the value based on the "best" value of $x_{1,1}$ to this point (and the remaining initial base values of our other decision variables). If our initial increase in $x_{1,1}$ has resulted in an improvement in the objective function, then we would not test for an improvement based on a decrease in the initial value of $x_{1,1}$. Instead, we would keep $x_{1,1}$ at its new increased value and test for improvements based on incrementing our second variable $x_{2,1}$.

With $x_{1,1}$ at its new "best" value and the objective function at its value based on $x_{1,1}$ and the initial base values of the remaining decision variables, we now increment $x_{2,1}$ and test for improvements in the objective function. We then hold $x_{2,1}$ at its new value ($x_{1,1}$ remains at its value found earlier) and increment the third decision variable. This process continues until all decision variables have been incremented. Once all variables have been incremented and tested for improvements, the new values of our decision variables \bar{x}_{new} provide an improvement in

our objective function as compared to our initial base values \bar{x}_{base} .

One method of continuing our search would be to let \bar{x}_{new} become our new base point and begin our step increments of each decision variable over again. However, we use a method developed by Hooke and Jeeves (Ref 15:168-169) instead to provide acceleration to the search procedure. We find a test point \bar{x}_{test} which is located in the direction of \bar{x}_{new} from \bar{x}_{base} , but twice the distance between \bar{x}_{new} and \bar{x}_{base} .

$$\bar{x}_{\text{test}} = 2\bar{x}_{\text{new}} - \bar{x}_{\text{base}}$$

We then increment each decision variable from its value in \bar{x}_{test} and attempt to find improvements in the objective function. We define \bar{x}_{new} as our base point and call it \bar{x}_{base} now. The new values of our decision variables as modified from (and reflecting improvements over) \bar{x}_{test} are now called \bar{x}_{new} . We then compare evaluations of the objective function based on \bar{x}_{new} and \bar{x}_{base} . If \bar{x}_{new} provides improvement in goal achievement over the goal achievement provided by \bar{x}_{base} , we continue to accelerate the search. (That is, $\bar{x}_{\text{test}} = 2\bar{x}_{\text{new}} - \bar{x}_{\text{base}}$, and we then define \bar{x}_{new} as our base point and call it \bar{x}_{base} .) If, however, \bar{x}_{new} provides no improvement in goal achievement over the goal achievement provided by \bar{x}_{base} , we return to \bar{x}_{base} and reduce our step size for each decision

variable. We then begin new, smaller increments of each decision variable from its value in \bar{x}_{base} . With the smaller step sizes it is possible that we may continue making improvements through searching and accelerating, or we may find that we need to reduce our step sizes even more.

The pattern search continues until one of the following events occurs:

1. We reach an optimum achievement of goals after which no improvements can be made (within certain prescribed tolerances for each priority level).
2. We terminate the search after reaching a predetermined maximum number of search patterns.
3. We terminate the search after reaching a predetermined maximum number of step size reductions.
4. We terminate the search due to computer computation time limitations.

Minor Modification of Problem Structure

Rather than continuing discussion of the pattern search procedure at this time, it is convenient to introduce a minor modification in our problem structure. With the present goal equations and objective function formulation we encounter difficulties when trying to support a rapid buildup of forces overseas. Because satisfying overseas requirements is at a higher priority level than satisfying CONUS requirements, overseas requirements are satisfied at all costs. Thus, it is possible for CONUS force levels

to be totally depleted while we attempt to satisfy extremely large overseas needs. This problem cannot be solved by simply exchanging the priority levels between the CONUS and overseas. Such an action would result in the CONUS needs being fully met, but with large underachievements of overseas goals.

The basic problem facing us is that we want to meet overseas requirements, but not seriously deplete our CONUS force. This problem is solved by introducing a new set of goal equations and a new preemptive priority level.

Our ninth goal equation set is concerned with maintaining minimum allowable CONUS force levels. The minimum levels represent a percentage of CONUS requirements $\bar{K}_{i,t}$. The equations are identical to the second equation set (People Available in the CONUS Base), except that now the CONUS goal $\bar{K}_{i,t}$ is multiplied by the minimum fraction of goal achievement authorized σ_i .

Ninth Equation Set--Minimum Allowable CONUS Levels.

$$\begin{aligned} & \sum_{j=1}^t (c_{i,j} + u_{i,j} + y_{i,j} - x_{i,j} - v_{i,j}) \\ & + d_{9,i,t}^- - d_{9,i,t}^+ = \sigma_i \bar{K}_{i,t} - K_{i,0} \\ & - (1 - \lambda_i) \sum_{j=1}^t w_{i,j-\Delta_i} \quad \forall i; t \leq \Delta_i \end{aligned} \quad (5.1)$$

$$\begin{aligned}
& (1 - \lambda_i) \sum_{j=\Delta_i+1}^t w_{i,j-\Delta_i} \\
& + \sum_{j=1}^t (c_{i,j} + u_{i,j} + y_{i,j} - x_{i,j} - v_{i,j}) \\
& + d_{9,i,t}^- - d_{9,i,t}^+ = \sigma_i \bar{K}_{i,t} - K_{i,0} \\
& - (1 - \lambda_i) \sum_{j=1}^{\Delta_i} w_{i,j-\Delta_i} \quad \forall i; t \geq \Delta_i \quad (5.2)
\end{aligned}$$

We introduce a new preemptive priority level in our objective function concerned with meeting minimum CONUS requirements. A reasonable place for the new priority level is between our present second and third levels. The new level becomes the third priority level with each remaining level being shifted down one unit.

<u>Old Priority Structure</u>	<u>New Priority Structure</u>
a_1	a_1
a_2	a_2
	a_3
a_3	a_4
a_4	a_5
a_5	a_6
a_6	a_7

$$a_3 = \sum_{t=1}^h \sum_{i=1}^n w_{9,i,t}^- d_{9,i,t}^- \quad (5.3)$$

Additionally, we make a minor change to our second priority level goals related to meeting desired trainee

per instructor ratios. We now express the desire that trainee per instructor ratios be met exactly, rather than simply not exceeded.

$$a_2 = \sum_{t=1}^h \sum_{j=1}^n (w_{6,i,t}^+ d_{6,i,t}^+ + w_{6,i,t}^- d_{6,i,t}^-) \quad (5.4)$$

This change allows added flexibility because we can return to our original second priority level structure by setting $w_{6,i,t}^- = 0 \quad \forall i,t$. However, unless otherwise stated, all weights in our objective function will be at the unit level. We now have an effective method for preventing unnecessary growth of the instructor pool and eliminating the generation of redundant alternate solutions.

General Comments About Pattern Search and Our Model

The pattern search method has been recommended by Ignizio as a useful approach for solving nonlinear goal programming problems (Ref 15:167). This is partly because a simplex procedure does not exist for the nonlinear case. Ignizio does state that the pattern search method cannot guarantee a solution which is a global optimum (Ref 15:171). Moreover, the procedure may converge to a local optimum and is dependent upon the initial guess for the decision variables. Depending upon the problem structure and the initial guess for \bar{x}_{base} , the solution procedure may exceed the number of patterns allowed or exceed the available computation time before converging to a solution.

Therefore, it is important that we thoroughly understand our problem structure and make a reasonable first guess for our decision variables. We also need to identify methods for avoiding the convergence of our search to a local optimum. Instead, we desire that our procedure converge to the true global optimum solution.

We are attempting to use a nonlinear procedure to solve a linear problem. Although we have a simplex procedure for linear problems, we are unable to use the simplex for realistic size formulations of our problem. This is due to computer memory and computation time limitations. However, by using the pattern search method, we are able to reduce considerably the amount of computer memory needed. While the simplex procedure requires computer space for a large simplex tableau (with many zero entries), the pattern search procedure requires space only for a small number of vectors (such as \bar{x}_{base} , \bar{x}_{new} , \bar{x}_{test}). The goal equations need not be accounted for in a large matrix as required by the simplex procedure. Instead, the values for the goal equations are generated into a vector \bar{y} by using FORTRAN arithmetic statements which take advantage of the recursive nature of the equations. Therefore, computer space requirements are relatively small for the pattern search method.

The computation time problem encountered while using the simplex procedure can also be alleviated to a large extent when using the pattern search method. In a large

size linear goal programming problem involving several decision variables and goal equations, a great deal of time is expended in performing the tableau manipulations necessary to find the global optimum solution. The pattern search method can also expend a great deal of time searching for a solution (and perhaps find only a local optimum). However, if we are familiar with our problem structure and can make a "good" initial guess for our decision variables, then we can reduce the amount of computation time needed considerably.

For example, it is reasonable to assume for our problem that the first three priority level goals will be fully satisfied. We know that the first priority level goals must be satisfied. Nothing should prevent the achievement of the second priority level goals related to maintaining desired trainee per instructor ratios. Likewise, our third priority level goals concerned with keeping the CONUS force within a certain percentage of requirements should also be met. However, we may not be able to satisfy the fourth priority level goals (overseas requirements) without also violating the achievement of our third priority level goals. Using such reasoning based on the characteristics of our problem structure, we can determine a reasonable first guess for our decision variables which satisfies several of the initial priority level goals. Hopefully, this guess will greatly reduce the search time needed for the problem.

While the simplex procedure has the disadvantages of requiring larger amounts of computer memory and computation time, simplex does have one distinct advantage over the pattern search. Given enough computer memory and computation time, simplex will find a global optimum solution (subject to possible roundoff error). However, the problem of converging to a local optimum when using the pattern search can be alleviated somewhat if we do three things.

1. Develop a thorough understanding of the problem structure.
2. Determine a "good" first guess for the decision variables based on the problem structure.
3. Modify the search procedure slightly to take advantage of the problem structure.

We now discuss the implications of attempting to solve our problem using the pattern search procedure discussed thus far in this chapter. The discussion demonstrates the need for accomplishing the three actions listed above.

For our problem structure the decision variables are listed in the following order, $\bar{x} = (x_{i,t}, c_{i,t}, v_{i,t}, y_{i,t}, u_{i,t}, w_{i,t}, e_t, v_{i,t})$, and this ordering affects how the search is accomplished. Let us assume that we decide to set our initial guess at the origin ($\bar{x}_{\text{base}} = (0,0,\dots,0)$), since this is where we begin our solution procedure using the simplex. We begin incrementing $x_{1,1}$, followed by $x_{2,1}$, and so forth, until we have incremented each $x_{i,t}$ variable by one step size. We then increment the c's, v's, y's,

u's, w's, and e's similarly. With each incrementation we try to improve goal achievement.

Let us assume that there is a requirement to build forces in the overseas theater (fourth priority level) and that our initial CONUS force begins at our CONUS requirements level (fifth priority level). Furthermore, we desire that the CONUS force remain at initial peacetime levels throughout the contingency. Additionally, we will allow some reductions in the CONUS force to within a certain prescribed percentage of our CONUS requirements (third priority level). With the above problem structure, movements of personnel from the CONUS to the overseas theater will result. ($x_{i,t}$ values will take positive steps from zero.) A small step increase in $x_{1,1}$ from $\bar{x}_{base} = (0,0,\dots,0)$ provides no change in the achievement of first, second, or third priority level goals. But, there is an improvement in achievement of the fourth priority level goals.

After the initial incrementing of the $x_{i,t}$ variables, we then increment the $c_{i,t}$ variables. A positive step for $c_{i,t}$ negates the overseas gains made by $x_{i,t}$, so this event is not allowed. A negative step for $c_{i,t}$ from zero results in a negative value for $c_{i,t}$, so this action is not permitted either. Initially, if the $x_{i,t}$ variables assume positive values, the $c_{i,t}$ variables will remain at zero.

The next variables to be incremented from zero values are the $v_{i,t}$ variables. There are no improvements in the achievement of any goals to be gained from allowing people to flow into the instructor pool from the CONUS. The instructor pool begins at its initial strength $I_{i,0}$, and no students so far are flowing into technical training ($w_{i,t} = 0$) $\forall i,t$. Moreover, removing people from the CONUS worsens the achievement of CONUS goals (fifth priority level) at this point. The $v_{i,t}$ variables will initially remain at zero.

Each $y_{i,t}$ variable takes one positive step so long as improvements result in the fifth priority level goal achievement and limitations on the availability of reserves (first priority level) are not exceeded.

Each $u_{i,t}$ variable takes a positive step within the limitations we have set on maintaining a stable instructor force (first priority level) so long as improvements are made in meeting CONUS requirements (fifth priority level). If the $y_{i,t}$ values have already ensured the satisfaction of CONUS requirements, then the $u_{i,t}$ variables remain at zero.

The next variables to be incremented from zero values are the $w_{i,t}$ variables. Each $w_{i,t}$ variable for $t \leq \tau + 1$ can be incremented positively as long as we stay within earlier gains into basic training before the contingency began (first priority level) and also do not worsen the meeting of desired trainee per instructor ratios (second

priority level). However, for $t \geq \tau + 2$, no further positive incrementations for the $w_{i,t}$ variables can result. Such actions result in violations of first priority level goals relating to earlier gains into basic training. We have not yet incremented the e_t variable from zero values. When $e_1 = 0$, then $w_{i,\tau+2}$ must also be zero.

The final variables to be incremented from zero values are the e_t variables. We cannot increment any e_t variable without violating first priority level goals. A positive value for e_1 implies that we also need a positive value for $w_{i,\tau+2}$ for at least one skill level i . However, we found above that $w_{i,t} = 0 \quad \forall t \geq \tau + 2; i = 1, \dots, n$.

In our second search pattern (after we have accelerated to \bar{x}_{test}), we notice for the most part the same variables increasing and the same variables unable to change. We need the capability of increasing the e_t and $w_{i,t}$ variables in order to find an optimum solution. This is because the labor pool is normally the greatest source of personnel resources available to help satisfy overseas and CONUS requirements. However, our problem structure and initial guess prevent us from taking advantage of our labor pool. Thus, any convergence is to a local optimum solution only.

In summary, we are unable to converge to a global optimum solution using the present pattern search procedure for the following reasons:

1. Our initial guess at the origin does not take advantage of the problem structure in that we cannot use the available civilian labor.

2. The search procedure allows incrementation of only one decision variable at a time, although simultaneous variable incrementations might be useful.

3. The sequence of our search is restricted to the ordering of our decision variables.

Modification of Search Procedure for Our Problem

In order that we may find an improved (and hopefully global optimal) solution to our problem, we need to modify both our initial guess for \bar{x}_{base} and our search procedure. We want to allow the realization of gains into basic training from the labor pool. While several possibilities exist for improvements in our solution procedure, we focus on one possibility below, which appears to be useful.

We first discuss an initial guess for \bar{x}_{base} which supports a change in the search procedure to be presented later. In our initial guess we now assign values to the $x_{i,t}$ and $c_{i,t}$ decision variables so that our overseas force requirements are met exactly. These values may be calculated directly from the first equation set (People Available in the Overseas Theater). As a result, the fourth and sixth priority level goals are totally satisfied. We assign values to the $y_{i,t}$ variables so that the reserves are activated into the CONUS force as rapidly as possible

within the reserve activation limitations. The first priority level goals relating to reserve activation are still fully satisfied.

We also make a change in the first priority level goals relating to the flow rate of people into technical training from basic training. We now include only the positive deviation variables from the fifth equation set in the first priority level of the objective function. This is equivalent to stating that our gains into technical training can go below, but not above, the amounts of people who successfully complete basic training. This is not realistic, because people who successfully complete basic training must enter one of the technical schools. However, the reason for this approach becomes apparent when the change in the pattern search method is discussed later.

As part of our initial guess for \bar{x}_{base} , we assign the maximum possible values to the e_t variables within the limitations of our civilian labor pool. The first priority level goals relating to civilian labor limitations and flow rate of people into technical training remain fully satisfied. Additionally, we assign values to $w_{i,1}$ $\forall i$ so that the trainee per instructor ratios are totally satisfied ($w_{i,1} = E_i I_{i,0}$) and gains into technical training equal the basic training completion rate ($\sum_{i=1}^n w_{i,1} = \beta e_{-1}$).

For reasons which become apparent later, we assign initial values to the $u_{i,1}$ variables so that all initial instructors $I_{i,0}$ are returned to the CONUS force during the time interval $(0,1)$. We also let $\rho = 1$ so that one hundred percent (100%) of our instructors may be returned to the CONUS any time. We set all remaining decision variables equal to zero.

Before discussing changes in the search procedure, it is useful to review our current extent of goal satisfaction based on our new initial guess for \bar{x}_{base} .

First Priority Level. All first priority level goals are fully satisfied.

Second Priority Level. All second priority level goals (trainee per instructor ratios) are fully satisfied.

Third Priority Level. Because we now meet our overseas goals exactly, we probably do not have our CONUS force at or above its predetermined percentage of CONUS requirements. Therefore, third priority level goals are not likely to be satisfied.

Fourth Priority Level. Overseas force requirements are completely satisfied.

Fifth Priority Level. CONUS force requirements are probably not satisfied.

Sixth Priority Level. Overseas requirements are not exceeded, so sixth priority level goals are completely satisfied.

Seventh Priority Level. Because we have activated the reserves as quickly as possible, our CONUS force probably exceeds CONUS requirements in the early periods of the contingency. (The opposite case is likely to be true in later periods.) Therefore, seventh priority level goals are not likely to be fully satisfied.

Our solution procedure is now modified in order to help avoid the problems we encountered earlier. Key aspects of the modification include selectively deciding which decision variables to iterate and determining useful combinations of variables to iterate simultaneously.

First Step. We now begin our search procedure by "freezing" all decision variables at their initial base values in \bar{x}_{base} , except for the $u_{i,t}$ and $w_{i,t}$ variables for $t \geq 2$ and the $v_{i,t}$ variables for $t \geq 1$.

Second Step. We now begin the simultaneous increments of $v_{1,1}$, $w_{1,2}$, and $u_{1,2}$. $w_{1,2}$ is incremented by its normal step size, but $v_{1,1}$ and $u_{1,2}$ are incremented by the step size of $w_{1,2}$ divided by the desired trainee per instructor ratio for skill level 1, E_1 . All instructors were initially returned to the CONUS during the time interval $(0,1)$ with our initial guess for $u_{1,1}$. Therefore, in order to bring any students into technical training during the time interval $(1,2)$ using $w_{1,2}$, we need to have a sufficient number of instructors available at $t = 1$. This is accomplished by moving just enough instructors

into the instructor pool using $v_{1,1}$ to satisfy our desired trainee per instructor ratio. Additionally, since we are not yet incrementing $w_{i,3}$ from 0, we also return all instructors to the CONUS during the time interval (1,2) using $u_{1,2}$. This avoids worsening the achievement of second priority level goals. While our second priority level goals remain fully satisfied, we realize improvements in our third and fifth priority levels (CONUS requirements) as a result of simultaneously incrementing $v_{1,1}$, $w_{1,2}$, and $u_{1,2}$.

After incrementing $v_{1,1}$, $w_{1,2}$, and $u_{1,2}$, we then simultaneously increment $v_{2,1}$, $w_{2,2}$, and $u_{2,2}$. This is followed by the combinations $v_{i,1}$, $w_{i,2}$, $u_{i,2}$ for $i = 3, \dots, n$. We then accomplish a second cycle of incrementations for $v_{i,1}$, $w_{i,2}$, $u_{i,2}$ $\forall i$, a third cycle, and so forth. We continue this process as long as we continue to make improvements in achieving CONUS goals within the limitations imposed by the labor pool and training pipeline (first priority level). Thus far in our modified search procedure, we have allowed only the decision variable combinations $v_{i,1}$, $w_{i,2}$, $u_{i,2}$ $\forall i$ to be incremented. One way to interpret the limited search procedure is to say that we are trying to satisfy CONUS goals as much as possible for the remainder of the contingency by the flow of trainees into technical training in the time interval (1,2).

After maximum goal satisfaction has resulted from the incrementations of $v_{i,1}$, $w_{i,2}$, and $u_{i,2}$ $\forall i$, we then freeze these variables in their present values. We then perform a similar simultaneous search procedure for $v_{i,2}$, $w_{i,3}$, and $u_{i,3}$ $\forall i$, followed by a search for $v_{i,3}$, $w_{i,4}$, and $u_{i,4}$ $\forall i$, and so forth. Our last simultaneous search is made for $v_{i,h-1}$, $w_{i,h}$, and $u_{i,h}$. Since we earlier set our values for e_t $\forall t$ to their maximum possible levels based on civilian labor limitations, we encounter no problems initially in incrementing our $w_{i,t}$ values. Since we no longer require that our gains into technical training at least equal our completion rate for basic training (first priority level), then we use the numbers of trainees entering technical training needed to provide improvements in our CONUS goals.

Third Step. After maximum goal satisfaction has resulted from the incrementations of the combinations of $v_{i,t}$, $w_{i,t+1}$, $u_{i,t+1}$ as discussed above, we now "freeze" all decision variables at their latest values. At this point we should notice that significant improvements have been made in meeting CONUS requirements (third and fifth priority levels). The first, second, fourth, and sixth priority level goals are still fully satisfied. Some worsening in the seventh priority level has probably resulted because we have further exceeded CONUS requirements during the initial phases of the contingency. Even though we have made improvements in meeting CONUS

requirements, we probably still have not fully met our prescribed minimum CONUS force levels (third priority level) throughout the contingency. There is an apparent conflict here, because at the same time all overseas theater force requirements have been met (fourth priority level).

Fourth Step. We now attempt to rectify the conflict between the third and fourth priority level goals. Our approach is to postpone the movement of people to the overseas theater from the CONUS until our CONUS force is at least at its minimum authorized levels.

The problem facing us is represented by Figure 5-1. There are no goal conflicts in the time intervals $(0, t_1)$ and $(t_2, 20)$ for the 20-period contingency represented. However, during the time interval (t_1, t_2) our CONUS force has dropped below its minimum authorized levels for skill level i because we have satisfied overseas requirements instead. To rectify this conflict, we need to insure that our actual CONUS levels never drop below the minimum authorized CONUS levels.

A method for solving this problem is to institute a search procedure which moves the actual CONUS force levels to the minimum authorized levels during the interval (t_1, t_2) . As a result we still have maximum achievement of overseas requirements within the limitations imposed by the third priority level.

We now begin our search by simultaneously incrementing $x_{1,1}$ by a negative step and $x_{1,2}$ by a positive step. This

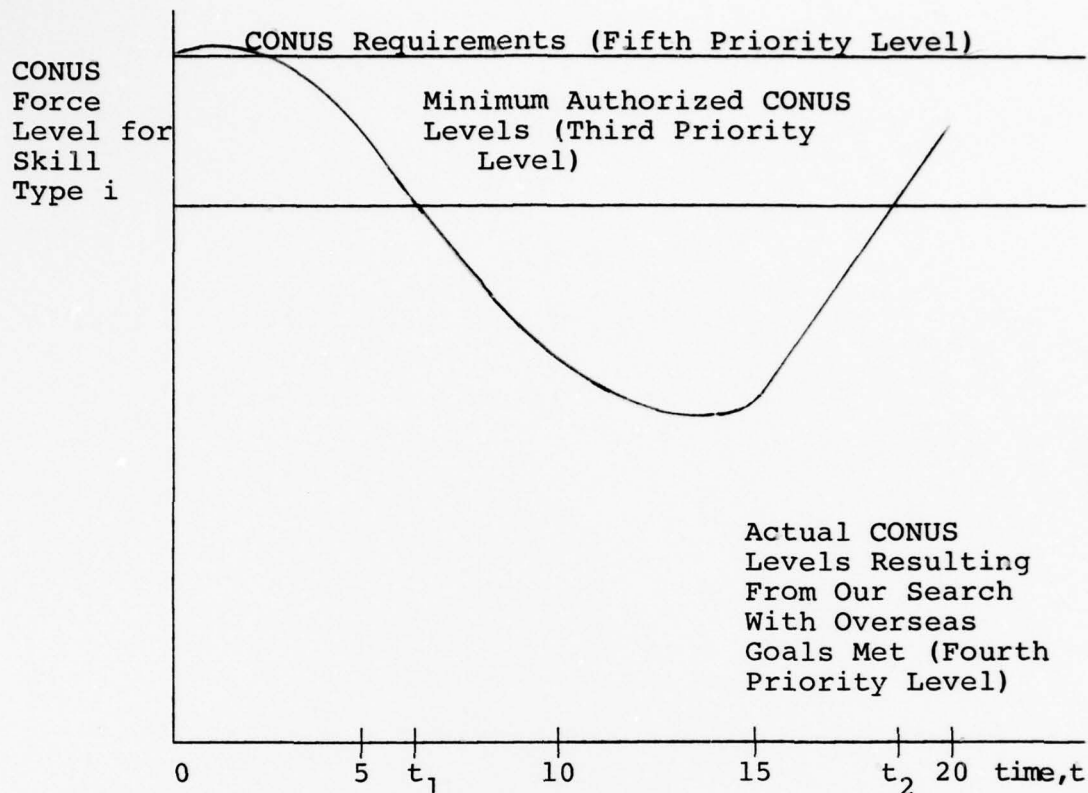


Figure 5-1. Conflict Between Achievement of Third and Fourth Priority Level Goals

procedure has a net effect of postponing the flow of personnel overseas while we attempt to improve third priority goals. Normally, no improvements will result for the $x_{1,1}$, $x_{1,2}$ combination. We attempt the same procedure for the remaining combinations $x_{i,1}$, $x_{i,2}$ for $i = 2, \dots, n$. When no improvement occurs, the variables are returned to their original values. We next attempt the same procedure for the $x_{i,2}$, $x_{i,3}$ combinations. Again, no improvements probably result. Let us assume that after time t_1 , the CONUS force for skill level 1 begins to drop below

minimum CONUS requirements. The simultaneous negative incrementation of x_{1,t_1+1} and positive incrementation of x_{1,t_1+2} will result in an improvement in the third priority level goals for time t_1+1 . At the same time no changes will occur in the third priority level for other time periods for skill level 1. A similar point in time is also reached for the other skill types after which improvements are made in the third priority level goals by the search procedure. There may also be a time later in the contingency t_2 when further improvements in the third priority goals are not required because minimum CONUS force levels are met.

The search procedure described above continues until each variable combination $x_{i,t}$, $x_{i,t+1}$ has been allowed to increment by a single step. The net result to this point will be an improvement in the achievement of minimum CONUS requirements and a worsening in achievement of overseas requirements. Referring to Figure 5-1, the actual CONUS level for skill type i has increased by a step size for $t_1 < t < t_2$. We have begun the movement of our actual CONUS levels to the minimum desired CONUS requirements for $t_1 < t < t_2$.

We now continue our search process by repeating the procedure described above until the third priority level goals are fully satisfied. For each skill type and time period in the interval (t_1, t_2) , we attempt to just satisfy

third priority level goals, so that we still meet overseas goals to the maximum extent.

Fifth Step. We now attempt an improvement in the achievement of seventh priority level goals relating to our desire not to exceed CONUS requirements. For those initial time periods in which the CONUS force is overstrength, we simply reduce the values of the $y_{i,t}$ variables accordingly and allow the differences in reserves to be activated later when needed.

Sixth Step. We now correct the problem of certain gains into basic training being in excess of the needs of our contingency. Rather than use the maximum values of e_t specified by our initial guess, we instead calculate the levels of e_t needed to provide the personnel flows into technical training found in our earlier search procedure.

Seventh Step. During the second step of our modified search procedure, we allowed frequent movements of instructors back and forth between the CONUS and instructor pools. We did this so that desired trainee per instructor ratios (second priority level) were met exactly. While such action was convenient for the search procedure, we do not want this resulting instructor instability reflected in our final solution. Thus we calculate net flows and assign all net flows to CONUS as $u_{i,t}$ variables and all net flows to the instructor pool as $v_{i,t}$ variables.

Our problem is now solved by our modified search procedure. While there is no guarantee that we have found a global optimum solution, we have found a solution that is both useful and reasonable. Our result is much better than the results found by the earlier pattern search method because we have taken advantage of the problem structure.

Pattern Search Computer Algorithm

Appendix C provides a listing of the pattern search computer algorithm. The algorithm is derived from a pattern search computer program provided by Ignizio (Ref 15: 228,231-234,242-247) and incorporates the modified search procedures discussed in this chapter.

The design of the pattern search algorithm is depicted in Figure 5-2, and the various subprograms and their related functions are described below:

PATSH	controls the initialization and execution phases of the algorithm (the MAIN program).
DATAIN	reads in the initial data and prepares the matrices for later computation.
HJALG	begins the pattern search and accelerates toward a solution if achievement of goals is possible.
VALUE	evaluates the objective function for improvement.
YVALUE	creates current values of the goal equations using the latest decision variables.

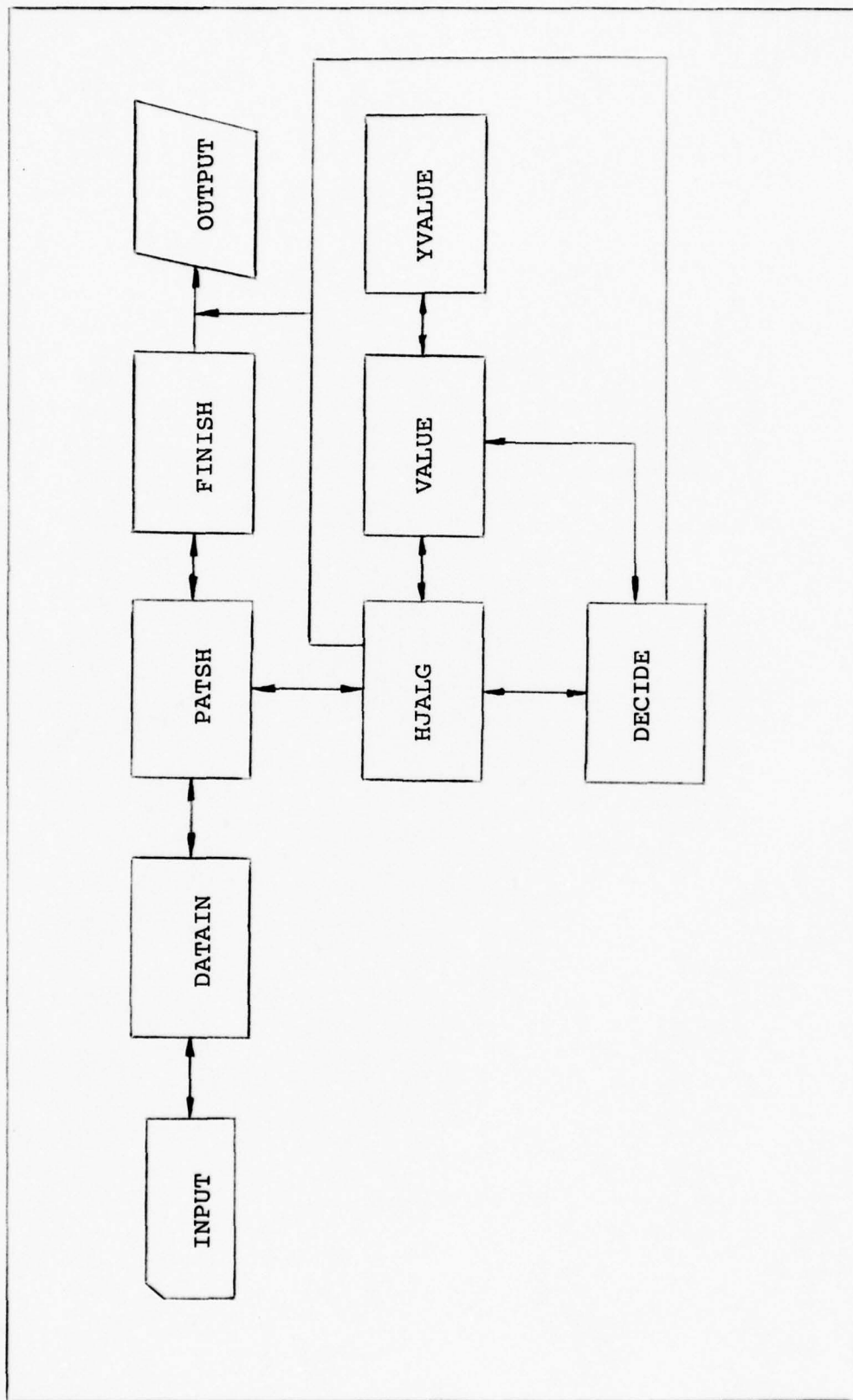


Figure 5-2. Pattern Search Design

DECIDE makes the actual decision whether or not an improved solution results from the last call to DECIDE.

FINISH stops the algorithm if the maximum number of pattern search cycles is reached.

Testing of the Modified Search Procedure

We now conduct testing of the modified search procedure using the computer algorithm for numerical problem formulations.

Test One. Our first test is based on a three-skill type, twenty-time period model. The problem formulation includes the additional set of goal equations and modification of the objective function established in this chapter.

Tables V-1 through V-4 summarize parameters and initial conditions for the problem. The three skill types considered are mission, direct mission support, and indirect mission support. One time period equals one month.

Weights of 100 are assigned to the negative deviation variables concerned with meeting CONUS requirements for skill type 2 for $6 \leq t \leq 20$ (fifth priority level). Weights of 10 are assigned to negative deviation variables concerned with meeting CONUS requirements for skill type 1 for $6 \leq t \leq 20$. It is anticipated that the required conscription rate should be fairly low near the end of the contingency. Therefore, zero weights are assigned to the negative deviation variables concerned with using all

Table V-1
Parameters and Initial Conditions

Skill Type, i	1	2	3
Initial CONUS Level, $K_{i,0}$ (thousands)	75	152	170
Minimum Fraction of CONUS Goal Achievement Authorized, σ_i	.8	.8	.8
Length of Technical Training, Δ_i (months)	4	6	2
Technical Training Attrition Rate, λ_i	0	0	0
Desired Trainee per Instructor Ratio, E_i	8	5	10
Initial Instructor Levels, (thousands) $I_{i,-1}$.209	.634	.333
$I_{i,0}$.209	.634	.333
Initial Instructor Flow Rates, (thousands) $v_{i,0}$	0	0	0
$u_{i,0}$	0	0	0
Initial Reserve Level, (thousands) $R_{i,0}$	30	30	30
Desired Residual Reserve, (thousands) \bar{R}_i	0	0	0
Reserves Activation Limitation, (thousands) $\bar{Y}_{i,1}$	10	10	10
$\bar{Y}_{i,2}$	20	20	20
$\bar{Y}_{i,t}; t \geq 3$	30	30	30
Initial Overseas Levels (thousands) $T_{i,0}$	25	38	30
Overseas Attrition Rates, $\alpha_{i,t} \quad 1 \leq t \leq 3$.03	.02	.02
$\alpha_{i,3}$.02	.013	.01
$\alpha_{i,t} \quad t \geq 4$.01	.005	.001

Table V-2
Initial Basic Training Pipeline
(in Thousands)

e_{-2}	e_{-1}	e_0
11.67	11.67	11.67

Table V-3
Initial Technical Training Pipeline
(in Thousands)

i	$w_{i,-5}$	$w_{i,-4}$	$w_{i,-3}$	$w_{i,-2}$	$w_{i,-1}$	$w_{i,0}$
1			1.67	1.67	1.67	1.67
2	3.17	3.17	3.17	3.17	3.17	3.17
3					3.33	3.33

Table V-4
Additional Parameters and Initial Conditions

Length of Basic Training, τ (months)	2
Basic Training Survival Rate, β	.7
Maximum Fraction of Instructors Permitted to Return to CONUS, ρ	1
Civilian Labor Available (thousands)	
	L_1 11.67
	$L_t, t \geq 2$ 50

available labor in the latter time periods of the contingency (first priority level). Unit weights are assigned to all other deviation variables in the objective function.

Tables V-5 through V-7 summarize results of the modified pattern search solution procedure for Test One. The first, second, and third priority level goals are fully satisfied. The sixth and seventh priority level values are close enough to zero so that we may consider the associated goals also effectively satisfied. However, neither overseas requirements nor CONUS requirements are fully met.

Table V-5
Extent of Goal Achievement for Test One

a_1	0
a_2	0
a_3	0
a_4	1526.08
a_5	45336.92
a_6	.40
a_7	1.75

Table V-6

Resulting CONUS and Overseas Levels of Test One (in Thousands)

t	i	Scenario			Solution Levels			t	i	Scenario			Solution Levels		
		CONUS	O/S	Requirements	CONUS	O/S				CONUS	O/S	Requirements	CONUS	O/S	
1	1	75	29	75	75	29		8	1	75	125	75	60.5	77.7	
	2	152	42	152	152	42			2	152	213	152	123.5	113.9	
	3	170	32	170	170	32			3	170	105	170	139	105	
2	1	75	33	75	75	33		9	1	75	140	75	61	87.2	
	2	152	60	152	152	60			2	152	238	152	122.5	119.4	
	3	170	35	170	170.7	35			3	170	120	170	136.1	110	
3	1	75	50	75	75	50		10	1	75	155	75	61.4	96.7	
	2	152	83	152	143.3	83			2	152	263	152	123.3	123	
	3	170	40	170	170	40			3	170	130	170	136.3	112	
4	1	75	70	75	61.9	70		11	1	75	170	75	61.7	104.3	
	2	152	113	152	122	103			2	152	272	152	123.3	148.7	
	3	170	45	170	170	45			3	170	131	170	136.7	113	
5	1	75	85	75	61.9	71		12	1	75	173	75	61.3	115.9	
	2	152	138	152	121.6	106.1			2	152	276	152	122.3	169.3	
	3	170	60	170	170	60			3	170	134	170	136	116	
6	1	75	95	75	61.6	73.1		13	1	75	173	75	62	126.5	
	2	152	163	152	123.1	107.2			2	152	276	152	122.6	191.8	
	3	170	75	170	164.7	75			3	170	134	170	136.9	118.1	
7	1	75	110	75	60.1	76.4		14	1	75	173	75	61	137	
	2	152	188	152	132.5	110.5			2	152	276	152	123.2	212.2	
	3	170	90	170	151.8	90			3	170	134	170	137.8	126.1	

Table V-6--Continued

T	i	Scenario Requirements			Solution Levels			t	i	Scenario Requirements			Solution Levels		
		CONUS	O/S		CONUS	O/S				CONUS	O/S		CONUS	O/S	
15	1	75	173		60	147.3		18	1	75	173		68	173.1	
	2	152	276		121.9	234.6			2	152	276		122	274.9	
	3	170	134		136.1	134			3	170	134		143.1	134	
16	1	75	173		61.9	169.6		19	1	75	173		66.3	173.1	
	2	152	276		122.5	254.8			2	152	276		127.9	276	
	3	170	134		136.1	134			3	170	134		148.3	134	
17	1	75	173		69.8	173.1		20	1	75	173		64.6	173.1	
	2	152	276		123.1	274.9			2	152	276		126.5	276	
	3	170	134		136.1	134			3	170	134		152.2	134	

t represents the time period.

i represents the skill type.

Table V-7

Resulting Flows and Instructor Levels for Test One (in Thousands)

t	i	$x_{i,t}$	$c_{i,t}$	$y_{i,t}$	$u_{i,t}$	$v_{i,t}$	$w_{i,t}$	e_t	$I_{i,t}$
1	1	4.75	0	3.19	.209	.313	1.67	11.67	.313
	2	4.76	0	1.97	.634	1.0	3.17		1.0
	3	2.6	0	0	.333	.067	3.33		.067
2	1	4.87	0	3.20	.313	.313	2.5	50.0	.313
	2	18.84	0	15.17	1.0	1.0	5.0		1.0
	3	3.64	0	0	.067	.067	.672		.067
3	1	17.99	0	16.31	.313	.313	2.5	50.0	.313
	2	24.2	0	12.36	1.0	1.0	5.0		1.0
	3	5.7	0	1.69	.067	.067	.672		.067
4	1	21	0	7.3	.313	1.344	2.5	50.0	1.344
	2	21.01	0	0	1.0	4.4	5.0		4.4
	3	5.4	0	4.89	.067	.225	.672		.225
5	1	1.7	0	0	1.344	1.344	10.75	50.0	1.344
	2	3.57	0	0	4.4	4.4	22.0		4.4
	3	15.05	0	14.38	.225	.225	2.25		.225
6	1	2.85	0	0	1.344	1.344	10.75	50.0	1.344
	2	1.69	0	0	4.4	4.4	22.0		4.4
	3	15.06	0	9.05	.225	.225	2.25		.225

Table V-7--Continued

t	i	$x_{i,t}$	$c_{i,t}$	$y_{i,t}$	$u_{i,t}$	$v_{i,t}$	$w_{i,t}$	e_t	$I_{i,t}$
7	1	3.95	0	0	1.344	1.344	10.75	50.0	1.344
	2	3.82	0	0	4.4	4.4	22.0		4.4
	3	15.08	0	0	.225	.225	2.25		.225
8	1	2.1	0	0	1.344	1.344	10.75	50.0	1.344
	2	3.94	0	0	4.4	4.4	22.0		4.4
	3	15.09	0	0	.225	.225	2.25		.225
9	1	10.25	0	0	1.344	1.344	10.75	50.0	1.344
	2	6.07	0	0	4.4	4.4	22.0		4.4
	3	5.12	0	0	.225	.225	2.25		.225
10	1	10.4	0	0	1.344	1.344	10.75	40.5	1.344
	2	4.19	0	0	4.4	4.4	22.0		4.4
	3	2.12	0	0	.225	.225	2.25		.225
11	1	8.55	0	0	1.344	3.203	10.75	0	3.203
	2	26.32	0	0	4.4	.05	22.0		.05
	3	1.13	0	0	.225	.913	2.25		.913
12	1	12.7	0	0	3.203	1.627	25.63	0	1.627
	2	21.36	0	0	.05	1.681	.25		1.681
	3	3.13	0	0	.913	.696	9.13		.696
13	1	11.73	0	0	1.627	0	13.02	9.60	0
	2	22.38	0	0	1.681	0	8.41		0
	3	2.13	0	0	.696	0	6.96		0

Table V-7--Continued

t	i	$x_{i,t}$	$c_{i,t}$	$y_{i,t}$	$u_{i,t}$	$v_{i,t}$	$w_{i,t}$	e_t	$I_{i,t}$
14	1	11.73	0	0	0	0	0	7.68	0
	2	21.38	0	0	0	0	0		0
	3	8.13	0	0	0	0	0		0
15	1	11.73	0	0	0	0	0	5.77	0
	2	23.38	0	0	0	0	0		0
	3	8.07	0	0	0	.672	0		.672
16	1	23.73	0	0	0	0	0	0	0
	2	21.33	0	0	0	0	0		0
	3	.13	0	0	.672	.538	6.72		.538
17	1	5.17	0	0	0	0	0	0	0
	2	21.4	0	0	0	0	0		0
	3	.13	0	0	.538	.404	5.38		.404
18	1	1.73	0	0	0	0	0	0	0
	2	1.38	0	0	0	0	0		0
	3	.13	0	0	.404	0	4.04		0
19	1	1.73	0	0	0	0	0	0	0
	2	2.51	0	0	0	0	0		0
	3	.13	0	0	0	0	0		0
20	1	1.73	0	0	0	0	0	0	0
	2	1.38	0	0	0	0	0		0
	3	.13	0	0	0	0	0		0

t represents the time period; i represents the skill type.

From Table V-6 it can be seen that, although CONUS force requirements are not fully achieved in most cases, the CONUS levels never fall below eighty percent (80%) of CONUS goals (third priority level). Also, CONUS levels exceed CONUS requirements only during the first two time periods for the third skill type. Moreover, these excesses are relatively small.

Because the CONUS levels must meet or exceed eighty percent (80%) of CONUS requirements, overseas theater requirements are not fully achieved for a large portion of the contingency. While overseas goals are effectively met during the first three and last four time periods, significant goal underachievements occur during the interim time periods. For example, at time $t = 12$ the following percentages of overseas goal achievement result.

Skill Type 1 - Sixty-seven percent (67%)

Skill Type 2 - Sixth-one and three-tenths percent (61.3%)

Skill Type 3 - Eighty-six and six-tenths percent (86.6%)

Skill type 3 (indirect mission support) has the best overall achievement of overseas goals, while skill type 2 (direct mission support) has the poorest overall achievement. This result has occurred even though higher weights were assigned to skill type 2 objective function terms and even though larger relative step sizes were assigned to skill type 2 decision variables during the pattern search. (During the pattern search, step sizes of 1,

.5, and .1 were assigned to $w_{1,t}$, $w_{2,t}$, and $w_{3,t}$, respectively.) The poor results for skill type 2 can be explained by at least four factors:

1. Skill type 2 has the largest overseas growth requirements.
2. Skill type 2 has the largest technical training time with the smallest trainee per instructor ratio.
3. The initial reserve force mix does not reflect the larger overseas growth requirements for skill type 2.
4. The initial CONUS force mix also does not reflect skill type 2 overseas growth needs.

The various flows shown in Table V-7 support the results found for the CONUS and overseas forces. Reserves are activated on an "as needed" basis, and gains into technical training reflect the relative step sizes used during the search procedure. Maximum utilization of available labor occurs during the first half of the contingency in an effort to build a wartime training pipeline. But, during the second half of the contingency, little use of civilian labor occurs. It seems reasonable to expect that significant gains into basic training should occur during the time period $t = 11$, because of the poor overall achievement of skill type 2 overseas goals during the middle time periods. However, such gains do not arrive in the CONUS force until time period $t = 20$.

Because we have not specified that all available labor be utilized during the latter time periods ($t \geq 16$), our

solution provides for only achievement of requirements during the span of the contingency. No provisions exist for anticipating requirements after the twentieth period. In order to properly anticipate training pipeline requirements to support time periods after the twentieth period, we would have to add additional structure to our model. This has not been accomplished in the interest of simplicity. An alternative approach for handling this problem would be to test the model for a figure like 30 time periods, but to use results for only the first 20 time periods. Thus, the training pipeline and instructor pool for the latter time periods of our 20-period contingency would be filled to levels required to support future needs during the immediate time after the twentieth period.

The values for the $u_{i,t}$ and $v_{i,t}$ variables result from the second step of the modified search procedure, and, thus, reflect highly unstable movements of the instructor force. This instability may be corrected by calculating and using the net flows between the CONUS and instructor pool. The frequent back and forth flows are only a solution technique for satisfying second priority level goals.

The present solution requires 55300₈ words of central memory storage, 8.776 CP seconds compilation time, 191.127 CP seconds execution time and 22.609 seconds input/output time.

Test Two. The model formulated for the first test is now evaluated under four additional scenarios. The overseas theater and CONUS requirements remain the same for each scenario, but different initial force mixes are now examined. Table V-8 provides specific conditions used for each scenario. Parameters, initial conditions and goals not specifically addressed remain unchanged from Test One.

Test 2A differs from Test One in that lower achievements of CONUS requirements are now authorized for skills 1 and 2 ($\sigma_1 = \sigma_2 = .7$).

Test 2B differs from Test 2A only in the initial training pipeline. While the total number of people initially in technical training is unchanged, larger numbers are now in skills 1 and 2 training. The new initial basic training level supports the technical training levels.

Test 2C differs from Test 2B only in the initial reserve force mix. While the total initial reserve force size is unchanged, larger portions of reserves are now in skills 1 and 2.

Test 2D differs from Test 2C only in the initial CONUS force mix. While the total initial CONUS force size is unchanged, larger numbers of people are now in skills 1 and 2. Initially the skill 3 CONUS level is at eighty-two and four-tenths percent (82.4%) of CONUS contingency requirements.

Table V-8

Testing Scenarios

Scenario	Test 1	Test 2 A	Test 2 B	Test 2 C	Test 2 D
Initial BMT Level (thousands) Σ (2 classes)	23.34	23.34	18.10	18.10	18.10
Initial Technical Training Level (thousands)					
Skill 1, Σ (4 classes)	6.68	6.68	8.68	8.68	8.68
Skill 2, Σ (6 classes)	19.02	19.02	23.02	23.02	23.02
Skill 3, Σ (2 classes)	6.66	6.66	.66	.66	.66
Initial Reserve (thousands)					
Skill 1	30	30	30	40	40
Skill 2	30	30	30	45	45
Skill 3	30	30	30	5	5
Initial CONUS Level (thousands)					
Skill 1	75	75	75	75	85
Skill 2	152	152	152	152	172
Skill 3	170	170	170	170	140
Initial Overseas Level (thousands)					
Skill 1	25	25	25	25	25
Skill 2	38	38	38	38	38
Skill 3	30	30	30	30	30
Minimum Permitted CONUS Levels (fraction of CONUS goals)					
Skill 1	.8	.7	.7	.7	.7
Skill 2	.8	.7	.7	.7	.7
Skill 3	.8	.8	.8	.8	.8

Table V-9 provides testing results under each scenario for the CONUS and overseas theater. (Even numbered periods are shown.)

It is reasonable that we find improvements in overseas goal achievement for skills 1 and 2 in Test 2A because lower CONUS levels for those skills are now authorized.

In Test 2B there are only moderate improvements in overseas goal achievement for skills 1 and 2 over Test 2A. There is also a somewhat poorer achievement of skill 3 overseas requirements. Redistribution of the initial technical training pipeline has little influence on improving the achievement of overseas goals.

Test 2C provides significant overseas improvements for skills 1 and 2 over Test 2B. The composition of the initial reserve force plays a key role in determining the outcome of our contingency. There is also a poorer achievement of skill 3 overseas requirements than found for Test 2B.

Similarly, Test 2D provides even better achievement of overseas needs for skills 1 and 2 than Test 2C. Thus, the composition of the initial CONUS force is also an influential factor for the contingency. The achievement of skill 3 overseas requirements continues to worsen.

Table V-10 summarizes the percentage achievement of force level requirements in periods 8 and 12.

Table V-9
Five Scenario Testing Results (in Thousands)

t	i	Requirements		Results Test 1		Results Test 2A		Results Test 2B		Results Test 2C		Results Test 2D	
		COONUS	O/S	COONUS	O/S	COONUS	O/S	COONUS	O/S	COONUS	O/S	COONUS	O/S
2	1	75	33	75	33	75	33	75	33	75	33	79.8	33
	2	152	60	152	60	152	60	152	60	152	60	156	60
	3	170	35	171	35	170.7	35	170	35	169.4	35	139.4	35
4	1	75	70	75	70	61.9	70	64	70	74	70	75	70
	2	152	113	143.3	103	112	113	114.8	113	124.8	113	144.8	113
	3	170	45	170	45	170	45	170	45	159	45	136	38
6	1	75	95	61.6	73.1	53.6	81.1	53.4	83	53.4	93	61.4	95
	2	152	163	123.1	107.2	107.1	123.1	107.3	127.1	108.3	141	108.3	161
	3	170	75	164.7	75	164.7	75	155.1	75	136.1	69	136.1	39
8	1	75	125	60.5	77.7	52.5	85.5	52.8	85.4	52.8	95.2	52.8	105.1
	2	152	213	123.5	113.9	107.5	129.6	107.3	133.5	108.3	147.3	108.3	167.1
	3	170	105	139	105	139	105	137.5	97	136.5	73	136.5	43.1
10	1	75	155	61.4	96.7	52.6	104.3	52.8	106.3	52.8	113.8	52.9	125.5
	2	152	263	123.3	123	106.8	140.5	108.3	142.4	107.7	156.1	106.5	179.7
	3	170	130	136.3	112	136.2	112	136.2	100	137.6	76.1	136.5	44.3
12	1	75	173	61.3	115.9	54.3	123.4	52.6	125.3	53.7	132.7	53.8	142.2
	2	152	276	122.3	169.3	108	184.7	108.3	184.6	106.8	202.1	106.7	221.4
	3	170	134	136	116	136.7	116	137.2	106.1	136.4	108.2	136.1	84.4

Table V-9--Continued

t	i	Requirements		Results Test 1		Results Test 2A		Results Test 2B		Results Test 2C		Results Test 2D	
		CONUS	O/S	CONUS	O/S	CONUS	O/S	CONUS	O/S	CONUS	O/S	CONUS	O/S
14	1	75	173	61	137	53	144.3	53.3	144.2	52.6	143.4	53.4	152.7
	2	152	276	123.2	212.2	107.2	227.5	107	229.4	108	242.7	108	261.9
	3	170	134	137.8	126.1	136.1	134	136.1	134	136.3	134	136.5	134
16	1	75	173	6.19	169.6	57.5	173	57.6	173	53.1	169.8	53.6	169.1
	2	152	276	122.5	254.8	106.5	269.9	108.3	269.7	107.8	268.9	113.2	276
	3	170	134	136.1	134	136.1	134	138	134	136.3	134	136.5	134
18	1	75	173	68	173.1	59.7	173	59.7	173	58.8	173	58.1	173
	2	152	276	122	274.9	113	276	106.7	274.8	112	276	110.5	276
	3	170	134	143.1	134	150.6	134	154.1	134	145	134	145.2	114
20	1	75	173	64.6	173.1	56.2	173	56.3	173	55.4	173	54.6	173
	2	152	276	126.5	276	110.3	276	110.2	276	109.4	276	108.4	276
	3	170	134	152.2	134	170.1	134	170.1	134	170.3	134	170.5	134

t represents the time period.

i represents the skill type.

Table V-10
Percent Force Level Achievement

t	i	Test 1		Test 2A		Test 2B		Test 2C		Test 2D	
		CONUS	O/S	CONUS	O/S	CONUS	O/S	CONUS	O/S	CONUS	O/S
8	1	80.7	62.2	70	68.4	70.4	68.3	70.4	76.2	70.4	84.1
	2	81.3	53.5	70.7	60.8	70.6	62.7	71.3	69.2	71.3	78.5
	3	81.8	100	81.8	100	80.9	92.4	80.3	69.5	80.3	41
12	1	81.7	67	72.4	71.3	70.1	72.4	71.6	76.7	71.7	82.2
	2	80.5	61.3	71.1	66.9	71.3	66.9	70.3	73.2	70.2	80.2
	3	80	86.6	80.4	86.6	80.7	79.2	80.2	80.7	80.1	63

t represents the time period.

i represents the skill type.

Test 2C appears to provide the best overall balance of CONUS and overseas forces for the three skill types. Better achievements for skills 1 and 2 result in Test 2D only at the added expense of quite low skill 3 overseas levels. In Test 2D some improvements in skill 3 overseas levels can occur if minimum CONUS levels for skill 3 are lowered. However, any reductions in minimum CONUS levels affect our ability to respond in a second theater, if required.

Moreover, the pattern search procedure appears to be a reasonable and useful solution algorithm for our model. It is quite capable of evaluating a series of different scenarios and demonstrating the advantages and disadvantages of each. Because it requires relatively small amounts of computer memory and computation time, the pattern search method should be useful for solving problems involving several job categories and time periods.

VI. Conclusions and Recommendations

Goal programming is a valuable analytic tool for handling a model with multidimensional goals. The objectives underlying this research have been to broaden our understanding of the art and science of goal programming and to develop a successful goal programming structure for our contingency model. These objectives have certainly been achieved. We hope that the work which is initiated with this study advances the usefulness and development of goal programming solutions to such problems.

The two approaches to problem solution in our research have strengths as well as weaknesses. Generally, a weakness of one method is a strength of the other. The weaknesses and strengths that we have encountered and considered relate to the usefulness, the conservation of available computer resources, and the applicability of the technique to the problem.

The simplex algorithm for solving linear goal programming problems is independent of the problem structure; its general nature poses few restrictions on the type of problem to which it can be applied. A fundamental strength for relying upon the simplex method is the provision of a "best" possible solution to a formal policy; furthermore, that solution is guaranteed to be "best"--the global

optimum over all optimal solutions. The algorithm examines every variable which is not in the current basis and considers it for possible entry into the basis at every iteration of every priority level.

The principal weakness of the simplex method is its use and accommodation of computer storage. The method requires large-scale matrices for many job categories and many time periods of interest. The position of coefficients within the technology matrix is important for the iterative procedure which is the essence of the method. As the number of job categories (n) or time periods (h) increases by one, the number of words of central memory required for the technology matrix increases by a polynomial of degree 3 in terms of both n and h . (With n job categories and h time periods, the number of decision variables equals $6nh + h$, and the number of constraint equations equals $5nh + 2h + n$. The total number of words required is the products of $(5nh + 2h + n)$ and $(6nh + h + 5nh + 2h + n)$ or $55n^2h^2 + 37nh^2 + 16n^2h + 5nh + 6h^2 + n^2$. If a job category is added to this model, the increase in words is $110nh^2 + 92h^2 + 32nh + 21h + 2n + 1$. If an extra time period is included, the increase in words is $110nh^2 + 71n^2 + 74nh + 42n + 12h + 6$.) The sparse nature of this technology matrix requires a minimum of forty-seven percent (47%) of all TE coefficients to be zero. Because of the storage requirements, the program has lengthy execution time and processing time for input and output.

Another weakness of the simplex algorithm is the inability to adapt to nonlinear constraints. The simplex method restricts all goal equations to be strictly linear; although our research is a development of linear equations, the adaptation of nonlinear equations to our contingency model is not permitted with the simplex method. Even logarithmic or other special transformations of the goal equations are meaningless in explaining goal deviations.

The pattern search technique offers several strengths. This technique yields a reasonable solution especially when the computer algorithm is tailored to fit the goal structure for the particular problem. The savings in the use of computer storage is evident, simply by the fact that zero values need not be kept for the goal equations. Computation time as well as input/output processing time is reduced for handling large cases. The "3 by 20" numerical example presented in the previous chapter finishes in less than 200 seconds of central processing time whereas the simplex version finishes in hours.

The weaknesses in pattern search stem from the flexibility to be tailored to a formal policy structure. The advanced search technique is totally dependent on the particular policy. This accommodation of policy to the design of the pattern search subroutines requires the analyst to examine every possible flow of personnel resources thoroughly. Without careful investigation of policy objectives and these flows of variables, the analyst discovers

that the computer program iterates and searches blindly, seeking improvement without guarantee of a mathematical global optimum. One final disadvantage of this approach is the sensitivity of the algorithm to incremental changes in step size. The computer program tries to find the minimal value of the objective function and depends heavily on the physical direction of the search in Euclidean n -space.

This research represents a first step in the development and implementation of a goal programming model for contingency planning. As a result, the model structure has remained relatively simple. Future enhancements can be made if one considers planning in other resource areas (for instance, materiel resources in lieu of personnel resources).

At this stage the pattern search procedure appears to provide results of an immediate and useful nature. Certainly, gains can be realized from future research involving this method. Moreover, even greater potential gains are possible from enhancement of the goal programming simplex method. By incorporating the notions of revised simplex procedures and a decomposition structure (Ref 29:224-285), it may be possible to realize significant improvements in computational efficiency while also finding a global optimum solution. Although Taha speaks to the linear programming context for revised simplex and decomposition

methods, research with these ideas in a goal programming framework may reduce the overhead of excessive storage and processing time and produce favorable results.

BIBLIOGRAPHY

Bibliography

1. Ackoff, Russel J., and Maurice W. Sasieni. Fundamentals of Operations Research. New York: John Wiley and Sons, Inc., 1968.
2. ASD Computer Center. CDC NOS/BE User's Guide, Revision E, January 1978.
3. Charnes, A., W. W. Cooper, D. Klingman, and R. J. Niehaus. Explicit Solutions in Convex Goal Programming. Management Sciences Research Report No. 341. Pittsburgh, Pennsylvania: Graduate School of Industrial Administration, Carnegie-Mellon University, June 1973. (AD 780791)
4. Charnes, A., W. W. Cooper, and R. J. Niehaus. A Goal Programming Model for Manpower Planning. Management Sciences Research Report No. 115. Pittsburgh, Pennsylvania: Graduate School of Industrial Administration, Carnegie-Mellon University, December 1967. (AD 664501)
5. Charnes, A., and W. W. Cooper. Goal Programming and Multiple Objective Optimizations (Part I). Research Report CCS 250. Austin, Texas: Center for Cybernetic Studies, the University of Texas, November 1975.
6. Charnes, A., W. W. Cooper, D. Klingman, and R. J. Niehaus. Some Advanced Start Procedures for Manpower Planning in Goal Programming Models. Management Sciences Research Report No. 342. Pittsburgh, Pennsylvania: Graduate School of Industrial Administration, Carnegie-Mellon University, March 30, 1974. (AD 780793)
7. Charnes, A., W. W. Cooper, and R. J. Niehaus. Studies in Manpower Planning. Washington, D.C.: U.S. Navy Office of Civilian Manpower Management, 1972.
8. CYBER Loader, Version 1, Reference Manual. Sunnyvale, California: Control Data Corporation, 1977, pp. 7-1 to 7-10.
9. Dantzig, George B. Linear Programming and Extensions. United States Air Force Project RAND Report. Princeton, New Jersey: Princeton University Press, 1963.
10. Dauer, Jerald P., and Robert J. Krueger. "An Iterative Approach to Goal Programming," Operations Research Quarterly, 28: 271-81 (1977).

11. FORTTRAN Extended, Version 4, Reference Manual. Sunnyvale, California: Control Data Corporation, 1977.
12. Goodman, S. E., and S. T. Hedetniemi. Introduction to the Design and Analysis of Algorithms. New York: McGraw-Hill, Inc., 1977.
13. Hillier, Frederick S., and Gerald J. Liebermann. Operations Research. San Francisco, California: Holden-Day, Inc., 1974.
14. Hooke, R., and T. A. Jeeves. "'Direct Search' Solution of Numerical and Statistical Problems," Journal of the Association for Computing Machinery, 8, 2:212-29 (April 1961).
15. Ignizio, James P. Goal Programming and Extensions. Lexington, Massachusetts: D. C. Heath and Company, 1976.
16. Ignizio, J. P., and D. E. Satterfield. Multicriteria Optimization in BMD Systems Design. National ORSA/TIMS meeting presentation, Atlanta, Georgia, 1977.
17. Ijiri, Y. Management Goals and Accounting for Control. Chicago: Rand-McNally, Inc., 1965.
18. Knight, Capt Jon M., Lt Col William H. Pope, and Capt Stanley B. Polk. Integrated Simulation Evaluation Model (ISEM) of the Air Force Manpower and Personnel System: Requirements and Concepts. Brooks Air Force Base, Texas: HQ Air Force Human Resources Laboratory, Occupation and Manpower Research Division, August, 1977. (AFHRL-TR-77-63)
19. Lee, Sang M. Goal Programming for Decision Analysis. Philadelphia: Auerbach Publishers, Inc., 1972.
20. Marini, Ronald J. Goal Programming. Master's Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1973. (AD 773037)
21. Niehaus, Richard J. Computer-Assisted Manpower Models Using Goal Programming. OCP Report No. 32. Washington, D.C.: U.S. Navy Office of Civilian Personnel, July 1977. (AD A045305)
22. Niehaus, D. J., D. Sholtz, and G. L. Thompson. Managerial Tests of Conversational Manpower Planning Models. OCMM Research Report No. 22. Washington, D.C.: Office of Civilian Manpower Management, Navy Department, April 1975. (AD A010825)
23. O'Brien, James A. Computers in Business Management. Homewood, Illinois: Richard D. Irwin, Inc., 1975.

24. Orchard-Hays, Wm. Background, Development and Extensions of the Revised Simplex Method. U.S. Air Force Project RAND Research Memorandum. Santa Monica, California: The Rand Corporation, April 30, 1954. (AD 90542)
25. "An Overview of the Prototype Integrated Simulation Evaluation Model of the Air Force Manpower and Personnel System." Contract No. F44620-76-C-0125. Pittsburgh, Pennsylvania: CONSAD Research Corporation, March 1, 1977.
26. Rosenbrock, H. H. "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer Journal, 3, 3:175-84 (October 1960).
27. Simmons, Donald M. Linear Programming for Operations Research. San Francisco, California: Holden-Day, Inc., 1972, pp. 89-158.
28. Stone, Harold S. Introduction to Computer Organization and Data Structures. New York: McGraw-Hill, Inc., 1972.
29. Taha, Hamdy A. Operations Research: An Introduction. New York: Macmillan Publishing Co., Inc., 1971, pp. 224-85.
30. Timar, John Joseph. Modelling, Transformations, and Scaling Decisions in Constrained Optimization Problems. Monterey, California: Naval Postgraduate School, March 1976. (AD A026396)
31. Turk, Daniel R. Priorities Modeling Using Goal Growth Programming. Rock Island, Illinois: Decision Models Directorate, Joint Conventional Ammunition Program Coordinating Group, August 1977. (AD A043867)
32. Welam, Ulf Peter. "Comments on Goal Programming for Aggregate Planning," Management Science, 22, 6:708-11 (February 1976).
33. Wilde, Douglas J., and Charles S. Breightler. Foundations of Optimization. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1967, pp. 271-344.
34. Wilde, Douglas J. Optimum Seeking Methods. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1964, pp. 145-57.
35. Yourdon, Edward. Design of On-line Computer Systems. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1972.

APPENDIX A
THE RECRUITING REQUIREMENTS MODEL

Appendix A. The Recruiting Requirements Model

The model attempts to meet manpower requirements by job category for a particular organization over several time periods and requires knowledge of the initial population per job category as well as total salary budget and total manpower capability. Provisions within the model account for hiring, firing, attrition, and individual movement among job categories. The model is expressed in words as shown in Figure A-1 (Ref 7:I-10).

The variables used in the mathematical formulation of the model are explained below (Ref 21:5).

- $E_k^+(t)$, $E_k^-(t)$ - positive or negative deviation, respectively, for the k^{th} manpower category in time t
- $x_k(t)$ - manpower on-board (in place) in the k^{th} manpower category in period t
- $y_k(t)$ - hires in the k^{th} manpower category in period t
- $z_k(t)$ - fires or reduction-in-force (RIFs) in the k^{th} manpower category in period t
- $G_k(t)$ - manpower requirement (goal) for the k^{th} manpower category in period t

OBJECTIVE: (a) MINIMIZE DISCREPANCIES FROM CIVILIAN MANPOWER REQUIREMENTS BY JOB CATEGORY
 (b) FAVOR ON-BOARD MANPOWER AND NEW HIRES OVER REDUCTIONS-IN-FORCE (RIF's)

SUBJECT TO:

NUMBER ON BOARD IN EACH JOB CATEGORY IN EACH PERIOD	- POSSIBLE AMOUNT OVER	+ POSSIBLE AMOUNT UNDER	- MANPOWER REQUIREMENTS BY JOB CATEGORY
NUMBER ON BOARD IN EACH JOB CATEGORY AT BASE PERIOD	- INITIAL POPULATION		
- NUMBER REMAINING IN EACH JOB CATEGORY	+ NUMBER ON BOARD IN EACH JOB CATEGORY AT ~ PRESENT PERIOD	- NEW + RIF's HIRES	- 0
SUM OF NUMBER ON BOARD IN EACH JOB CATEGORY	X	SALARY OF JOB CATEGORY	TOTAL SALARY BUDGET
SUM OF NUMBER ON BOARD IN EACH JOB CATEGORY	≤ TOTAL MANPOWER AVAILABLE		

Figure A-1. Recruiting Requirements Model

α_{kt}	- the weight which is applicable to the positive goal discrepancy for the k^{th} manpower category in period t
β_{kt}	- the weight which is applicable to the negative goal discrepancy for the k^{th} manpower category in period t
γ_{kt}	- the weight on the hires in the k^{th} manpower category in period t
δ_{kt}	- the weight on the fires or RIFs in the k^{th} manpower category in period t
M	- Markov-like transition matrix depicting the probabilities of internal personnel movement (m_{ij} is the fraction of personnel in job category j who move to category i after one period)
a	- vector of initial inventory of personnel in all job categories
e^T	- sum vector (i.e., $e^T = [1, 1, \dots, 1]$)
$C(t)$	- manpower ceiling for period t
$s^T(t)$	- average salary vector in period t
$B(t)$	- salary budgetary ceiling for period t

The model may now be expressed mathematically (Ref 21:6). (The objective is to minimize the weighted sum of deviations from gross manpower goals and weighted number of hires and fires (RIFs).)

$$\text{Minimize } \sum_k \sum_t \left[\alpha_{kt} E_k^+(t) + \beta_{kt} E_k^-(t) + \gamma_{kt} y_{kt} + \delta_{kt} z_k(t) \right]$$

Subject to

$$\text{(Goal Constraints)} \quad x_k(t) - E_k^+(t) + E_k^-(t) = G_k(t)$$

$$\text{(Manpower Transition Conditions)} \quad x(0) = a$$

$$-[Mx(t-1)]_k + x_k(t) - y_k(t) + z_k(t) = 0$$

$$\text{(Manpower Ceiling Constraints)} \quad e^T x(t) \leq C(t)$$

$$\text{(Salary Budget Constraints)} \quad s^T(t) x(t) \leq B(t)$$

(Nonnegativity Constraints)

$$x_k(t), y_k(t), z_k(t), E_k^+(t), E_k^-(t) \geq 0$$

The objectives of the model are accomplished by using relative weighting factors. Weights are set so that a penalty is paid whenever the manpower requirement (goal for each job category) is not met. A penalty is also paid for hiring or firing. The weighting factors may be adjusted so that on-board manpower and new hires are favored over excess personnel and RIFs. A given set of manpower requirements need only be met as closely as possible, and an increasing penalty (size of the objective function) is paid as one moves away from the goals. Other planning policies may be examined by changing the weighting scheme (Ref 7:I-9).

This model differs somewhat from the example of a typical goal programming problem presented earlier in that different preemptive priority levels are not shown. Instead, only relative weighting of goals is provided.

If weights used differ by large enough amounts, then solutions similar to those obtained by using preemptive priorities will result. However, in general, the use of preemptive priorities and weighting is preferred to the use of weighting alone in order to insure that higher priority goals are attained first and that satisfaction of lower goals will not violate already attained higher goals. Deviation variables are not provided in the manpower ceiling constraints and salary budget constraints. In the manpower transition condition equation, the hires and fires variables may be considered as deviation variables. Since there is only one priority level, the problem can be solved with linear programming.

APPENDIX B
THE SIMPLEX COMPUTER ALGORITHM

Appendix B. The Simplex Computer Algorithm

The computer programs that follow are the updated version of the original design (before the use of core-swapping) and the successor to that version in which core-swapping of the technology matrix is handled. These programs were designed to run on the CDC 6600 computer system associated with the Aeronautical Systems Division, Wright-Patterson Air Force Base, Ohio. The version of FORTRAN Extended, Version IV is applicable to that machine. Thus, the user of these codes ought to be aware of the nonstandard FORTRAN features employed:

1. The use of the alternate return on subroutine CALLs. This feature allows the program to return control to a statement label in the calling program unit if conditions within the subroutine warrant an alternate return rather than a normal return to the statement following the CALL.
2. The use of the STOP "cc...c" statement where cc...c is an alphabetic string of 1-70 characters.
3. The use of list-directed read and print statements in which no FORMAT is referenced.
4. The use of more than one entry point to a subroutine with the ENTRY statement (notice ENTRY PERM in SUBROUTINE TEST and ENTRY ERRORS in SUBROUTINE POUT).

5. The use of BUFFER IN and BUFFER OUT statements in the core-swapping algorithms for the top stub values.

6. The use of multiple equal signs for one statement as $N = NBLK = 0$. Otherwise, this statement would be written as two:

```
NBLK = 0  
N = NBLK
```

7. The use of the SECOND and EOF functions in order to determine the current system time and to test for an end-of-file mark on a file, respectively.

The first main program deals with the three-job-skill, six-time-period case described in Chapter Four. The COMMON block dimensions should be enlarged accordingly for problems dealing with more skills or more time periods. The remaining three main programs utilize the core-swapping algorithm for a three-job-skill, twenty-time-period model and should be run in the order presented in the appendix.


```

C      2 CALL INIT(NPRB), RETURNS(30)
C
C      SUBROUTINE INIT INITIALIZES THE INPUT FILES ACCORDINGLY.
C      FAULTY OR MISPUNCHED DATA CAN CAUSE ALTERNATE RETURNS
C      TO STATEMENT 30.
C
C      NPRT=0
C
C      THE DO 3 LOOP BEGINS THE MAJOR PORTION OF THE ALGORITHM.
C      WHEN NROW BECOMES NPRI + 1, THE ALGORITHM FOR DETER-
C      MINING THE FINAL SOLUTION IS ENDED. CALLING CINDX
C      CAUSES THE ROWS TO BE COMPUTED. SUBROUTINE TEST
C      THE ENTERING VARIABLE COLUMN (NEVC) AND THE DEPARTING
C      VARIABLE ROW (NDVR). WHEN NEVC=0, THE ROW CANNOT BE
C      OPTIMIZED FURTHER.
C
C      DO 4 NROW=1,NPRI
C      CALL CINDX(NROW,NROW)
C      CALL TEST(NEVC,NDVR,NROW), RETURNS(20)
C      DO 3 IVAL=1,IMAX
C      IF(NEVC.LE.0) GO TO 4
C
C      SUBROUTINE PERM COMPUTES THE NEW TABLEAU.
C
C      CALL PERM(NEVC,NDVR,NROW)
C      3 CALL TEST(NEVC,NDVR,NROW), RETURNS(20)
C      CALL ERRORS(IMAX)
C      GO TO 2
C      4 CONTINUE
C      NROW=NPRI
C
C      POUT IS CALLED TO PRINT A FINAL SOLUTION. IF THERE
C      ARE ANY ALTERNATE SOLUTIONS IN THE PROBLEM, SUBROUTINE
C      ALTST TESTS FOR VALID ONES AND PRINTS OUT ANY THAT
C      IT PRODUCES.

```

C
CALL POUT(NPRT)
CALL ALTST(NPRT,NROW)
GO TO 2
20 STOP "FAILURE"
30 STOP
END


```

C
C
C
C
C
C
C
C
SUBROUTINE INIT(NPRP), RETURNS(ERR1)

      THIS ROUTINE ESTABLISHES THE INITIALIZATION OF THE
      ALGORITHM BY READING IN THE INPUT DATA AND USING THIS
      DATA TO CALCULATE THE INITIAL COEFFICIENTS FOR THE
      APPROPRIATE MATRICES.

      COMMON TL(106,5), TT(6,220), TE(106,220), YI(6,220),
1      TR(106), TA(6), JCOL(220,2), JROW(106,2), X1(114,2),
2      NOBJ,NPRI,NVAR,NCOL
      COMMON /PAR/ L
      DIMENSION IPRI(4), ISUB(4), WHT=(4)
      READ(5,*) NOBJ,NPRI,NVAR,NTAF,L

      L IS A POSITIVE INTEGER IN THE RANGE 2,3,...,8
      USED FOR DECIMAL PRECISION

      IF(EOF(5).NE.0) RETURN ERR1
      IF(NOBJ.LT.1.OR.NOBJ.GT.106) GO TO 22
      IF(NPRI.LT.1.OR.NPRI.GT.6) GO TO 22
      IF(NVAR.LT.1.OR.NVAR.GT.220) GO TO 22
      NCOL=NOBJ+NVAR
      DO 1 NV=1,NVAR
      JCOL(NV,1)=2
1  JCOL(NV,2)=NV
      DO 2 NO=1,NOBJ
      NC=NO+NVAR
      JCOL(NC,1)=3
      JFOW(NO,1)=4
2  JCOL(NC,2)=JROW(NO,2)=NO
      DO 3 NV=1,NVAR
      DO 3 NO=1,NOBJ
3  TE(NO,NV)=0.
      READ(5,*) (TB(K),K=1,NOBJ)

      SUBROUTINE COEF DEVELOPS THE TE COEFFICIENTS OF THE
C
C

```

C C

INITIAL NONBASIC VARIABLES.

```

CALL COEF
DO 4 NOR=1,NORJ
DO 4 NO=1,NORJ
NCC=NO+NVAR
TE(NOR,NOC)=0.
IF(NO.EQ.NOR) TE(NOR,NOC)=-1.
4 CONTINUE
DO 6 NP=1,NPRI
DO 5 NO=1,NORJ
5 TL(NO,NP)=0.
DO 6 NC=1,NCOL
6 TT(NP,NC)=0.
I=MOD(NTAF,4)
IF(I.EQ.0) NO=NTAF/4
IF(I.NE.0) NO=NTAF/4+1
DO 7 NT=1,NO
READ(5,*) KK,(IPRI(K),ISUB(K),HTF(K),K=1,KK)
DO 7 K1=1,KK

```

C C C

SUBROUTINE PLACE INSERTS VALUES INTO STUBS.

```

7 CALL PLACE(IPRI(K1),ISUB(K1),HTF(K1)),RETURNS(40)
READ(5,35) ((X1(I,J),J=1,2),I=1,NVAR)
NPRB=NPRB+1
WRITE(6,36) NPRB
RETURN
22 WRITE(6,28)
28 FORMAT(1H " *INPUT VARIABLE EXCEEDS ALLOWABLE DIMENSION RANGE**")
35 FORMAT(8A10)
36 FORMAT(1H1///" PROBLEM ",I4," READ IN SUCCESSFULLY")
40 RETURN ERR1
END

```



```

TE(I,J+NEW)=-TE(I,J)
IF(I.EQ.J+J0RCAT) TR(I)=TB(I)-TH0(J2)*GAM(J2,1)*TE(I,J2)
GO TO 14
13 TE(I,J)=1.
TE(I,J+NEW)=-1.
14 CONTINUE
C
C CONSTRAINT TYPE II: CONUS AVAILABILITY
C
C READ(5,*) (TH0(K),DELI(K),K=1,J0RCAT)
C
C ALL OF THE LAMBDA VALUES ARE IN TH0.
C
DO 15 K=1,J0RCAT
15 TH0(K)=1.-TH0(K)
NEW2=NEW+NEW
DO 18 J=1,NEW
J2=J+NEW
NEXTC=NEW*5+J
II=0
DO 18 I=J2,NEW2,J0RCAT
IF(II.GT.0) GO TO 16
I2=MOD(I,J0RCAT)
IF(I2.EQ.0) I2=J0RCAT
16 II=II+1
IF(II.LE.DELI(I2)) GO TO 17
TE(I,NEXTC)=TH0(I2)
17 TE(I,J)=TE(I,J+NEW2)=-1.
18 TE(I,J2)=TE(I,J2+NEW2)=TE(I,J+NEW2*2)=1.
IF(SWITCH.EQ.0.) GO TO 200
NEXTR=NEW+1
DO 150 J=1,NVAR
DO 150 I=NEXT2,NEW2
150 TE(I,J)=-TE(I,J)
200 CONTINUE
C

```

C WORK1 HOLDS THE INITIAL K(I) VALUES AND ALFA HOLDS THE
C INITIAL W(I) VALUES.
C

```

DO 19 K=1,J03CAT
  K1=DELI(K)
  READ(5,*)(WORK1(K),(ALFA(K,J),J=1,K1))
  DO 19 J=2,K1
19 ALFA(K,J)=ALFA(K,J)+ALFA(K,J-1)
  J=DELI(1)
  DO 20 I=2,J03CAT
    IF(J.LT.DELI(I))J=DELI(I)
20 CONTINUE
  DO 21 I=1,J03CAT
    I3=DELI(I)
    DO 21 K=I3,J
21 ALFA(I,K)=ALFA(I,I3)
  DO 22 I=1,NEW
    I2=MOD(I,J03CAT)
    IF(I2.EQ.0)I2=J03CAT
    I3=(I-1)/J03CAT+1
    IF(I3.GT.J)I3=J
22 TB(NEW+I)=TB(NEW+I)-WORK1(I2)-TH0(I2)*ALFA(I2,I3)
    IF(SWITCH.EQ.0.)GO TO 400
  DO 350 I=NEXT2,NEW2
350 TB(I)=-TB(I)
400 CONTINUE

```

C CONSTRAINT TYPE III AND IV: MOVEMENT FROM THE RESERVES
C
C

```

NEW3=NEW+NEW2
NEW4=NEW2+J03CAT
DO 23 I=1,J03CAT
  DO 23 J=I,NEW,J03CAT
23 TE(NEW2+I,NEW3+J)=1.
  DO 24 I=1,NEW
24 TE(NEW4+I,NEW3+I)=1.

```



```

C C C      CONSTRAINT TYPE V: INPUTS INTO TECHNICAL TRAINING
C
C
C      NEXTR=NEW4+NEW
C      NEXTC=NEW3+NEW2
C      NEW6=NEW3+NEW3
C
C      READ LAG, BETA, WORK1(I) FOR I EQUAL 1 TO LAG + 1.
C      LAG IS THE TIME TO FINISH BASIC TRAINING (AN INTEGER
C      NUMBER OF PERIODS). BETA IS THE SURVIVAL FACTOR OF
C      THAT TRAINING. WORK1(I) HOLDS THE INITIAL VALUES
C      OF E (THAT IS, E(-1) AND E(0) FOR A LAG OF 1),
C      RESPECTIVELY.
C
C      READ(5,*) LAG,BETA
C      LAG1=LAG+1
C      READ(5,*) (WORK1(I),I=1,LAG1)
C      DO 26 J=1,JTIMES
C      DO 25 I=1,JOBSCAT
C      25 TF(NEXTR+J,NEXTC+I)=1.
C      IF(J.GT.LAG1) TE(NEXTR+J,NEW6+J-LAG1)=-BETA
C      IF(J.LE.LAG1) TB(NEXTR+J)=WORK1(J)*BETA
C      26 NEXTC=NEXTC+JOBSCAT
C
C C C      CONSTRAINT TYPE VI: TECHNICAL TRAINING INSTRUCTORS
C
C      NEXTR=NEXTR+JTIMES
C      NEXTC=NEW3+NEW2
C
C      READ IN THE APPLICABLE VALUES AS FOLLOWS--
C      CAPITAL E(I) INTO TH0, CAPITAL I(-1) INTO WORK1,
C      CAPITAL I(0) INTO WORK2, INITIAL V(I) INTO ALFA(I,1)
C      INITIAL U(I) INTO ALFA(I,2).
C
C      READ(5,*) (TH0(K),WORK1(K),WORK2(K),K=1,JOBSCAT),((ALFA(I,J),
C      1 J=1,2),I=1,JOBSCAT)

```

```

DO 28 J=1,NEW
NEXTC=NEXTC+1
DO 28 I=J,NEW,JOB CAT
IF(I.EQ.J) GO TO 27
I2=MOD(I,JOB CAT)
IF(I2.EQ.0) I2=JOB CAT
TE(NEXTTR+I,NEXTC-NEW)=TH0(I2)
TE(NEXTTR+I,NEXTC-NEW3)=-TH0(I2)
IF(I.EQ.J+JOB CAT) TR(NEXTTR+I)=TH0(I2)*WORK2(I2)
GO TO 28
27 TE(NEXTTR+I,NEXTC)=1.
IF(J.LE.JOB CAT) TR(NEXTTR+J)=TH0(J)*(WORK1(J)+ALFA(J,1)-ALFA(J,2))
28 CONTINUE

```

C
C
C

CONSTRAINT TYPE VII: FLOWS INTO BASIC TRAINING

```

NEXTTR=NEXTTR+NEW
DO 29 J=1,JTIMES
29 TE(NEXTTR+J,NEW5+J)=1.

```

C
C
C

CONSTRAINT TYPE VIII: INSTRUCTOR FLOWS TO CONUS

```

RFAD(5,*) PERCT
NEXTTR=NEXTTR+JTIMES
NEXTC=NEW3+NEW
DO 31 J=1,NEW
NEXTC=NEXTC+1
DO 31 I=J,NEW,JOB CAT
IF(I.EQ.J) GO TO 30
I2=MOD(I,JOB CAT)
IF(I2.EQ.0) I2=JOB CAT
TE(NEXTTR+I,NEXTC)=PERCT
TE(NEXTTR+I,NEXTC-NEW2)=-PERCT
IF(I.EQ.J+JOB CAT) TR(NEXTTR+I)=PERCT*WORK2(I2)
GO TO 31
30 TE(NEXTTR+I,NEXTC)=1.

```

```
IF(J.LE.JOBCAT) T9(NEXTR+J)=PERCT*(WORK1(J)+ALFA(J,1)-ALFA(J,2))
31 CONTINUE
   RETURN
   END
```

```

C SUBROUTINE PLACE(JPRI,JSUB,WHT),RETURNS(ERR1)
C
C SUPROUTINE PLACE POSITIONS THE ACHIEVEMENT FUNCTION
C VALUES IN LEFT HAND AND TOP STUBS.
C
C JPRI PRIORITY LEVEL
C JSUB +JSUB = P SUB JSJ3
C      -JSUB = N SUB JSJ3
C WHT WEIGHTING FACTOR
C
C COMMON TL(106,5),TT(6,220),TE(105,220),TI(6,220),
1 TR(106),TA(6),JCOL(220,2),JROW(106,2),
2 X1(114,2),NORJ,NPRI,NVAR,VCOL
C
C INPUT DATA IS CHECKED TO BE CERTAIN THAT IT IS IN
C RANGE.
C
C IF(JPRI.LT.1.OR.JPRI.GT.NPRI.OR.JSUB.EQ.0.OR.IABS(JSUB).GT.NOBJ)
1 GO TO 2
C
C POSITIVE VALUES OF JSUB INDICATE TOP STUB VALUES.
C
C IF(JSUB.GT.0) GO TO 1
C
C PLACE VALUE IN LEFT STUB
C
C JSUB= -JSUB
C
C OVERWRITES ARE DETECTED AS BEING AN ERROR.
C
C IF(TL(JSUB,JPRI).NE.0) GO TO 3
C TL(JSUB,JPRI) = WHT
C RETURN
1 ICOL= JSUB + NVAR
C
C PLACE VALUE IN RIGHT STUB
C

```

C

```
IF(TT(JPRI,ICOL).NE.0) GO TO 3
TT(JPRI,ICOL) = WHT
RETURN
2 WRITE(6,4)
GO TO 6
3 WRITE(6,5)
4 FORMAT(//, "***SUBSCRIPT OUT-OF-RANGE WHILE READING SUBSCRIPTS**")
5 FORMAT(//, "***OVERWRITE ATTEMPTED IN A STUB**")
6 RETURN ERR1
END
```



```

C
C      SUBROUTINE CINDX(I,NROW)
C
C      THE FUNCTION OF CINDX IS TO COMPUTE INDEX ROWS.
C      CINDX DOES THIS FROM ROW I TO ROW NROW.
C
COMMON TL(106,6), TT(6,220), TE(106,220), TI(6,220),
1      TB(106), TA(6), JCOL(220,2), JROW(106,2),
2      X1(114,2), NORJ, NPRI, NVAR, NCOL
DO 2 NP=I,NROW
TA(NP)=0.
C
C      THE RIGHT HAND SIDE OF THE INDEX ROWS ARE COMPUTED.
C
DO 1 NO=1,NORJ
1 TA(NP)=TA(NP)+TB(NO)*TL(NO,NP)
DO 2 NC=1,NCOL
TI(NP,NC)=-TI(NP,NC)
DO 2 NO=1,NORJ
TI(NP,NC)=TI(NP,NC)+TE(NO,NC)*TL(NO,NP)
2 CONTINUE
RETURN
END

```

AD-A065 909

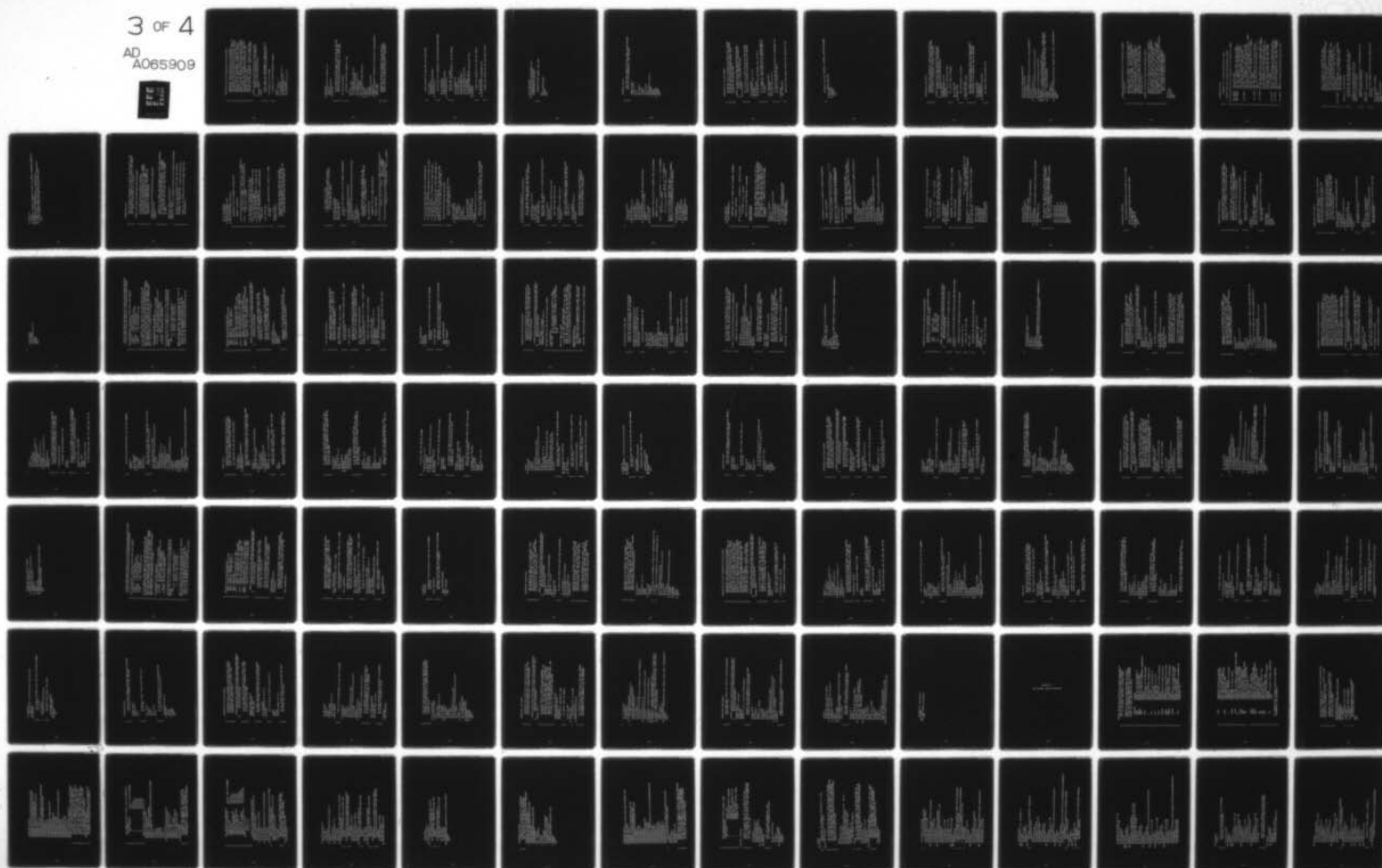
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/6 12/2
PERSONNEL CONTINGENCY PLANNING MODEL USING GOAL PROGRAMMING.(U)
DEC 78 J A MORENO, B W UTZ
AFIT/60R/SM/78D-10

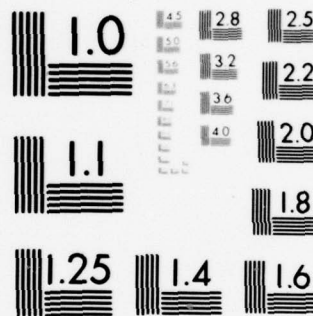
UNCLASSIFIED

NL

3 OF 4

AD
A065909





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A


```

1 CONTINUE
2 IF(TI(NROW,NC).LE.VEVC)GO TO 3
   NEVC=NC
   VFVC=TI(NROW,NC)
3 CONTINUE

   IF NO POSITIVE VALUES EXIST WHICH HAVE NO NEGATIVE
   INDEX VALUE ABOVE - RETURN VEVC = 0, SINCE NO
   FURTHER OPTIMIZATION IS POSSIBLE AT THIS LEVEL.

   IF(NEVC.EQ.0)RETURN

   DETERMINE DEPARTING VARIABLE'S ROW

   NDVR=0
   DO 7 NR=1,NORJ
   IF(TE(NR,NEVC).LE.0)GO TO 7
   V = TB(NR)/TE(NR,NEVC)
   IF(NDVR.EQ.0) GO TO 6
   IF(V-VDVR)5,4,7
4 DO 5 NP=1,NPRI
   IF(TL(NR,NP)-TL(NDVR,NP))7,5,5
5 CONTINUE
6 VDVR=V
   NDVR=NR
7 CONTINUE
   IF(NDVR.GE.1)RETURN
   WRITE(6,8)
8 FORMAT('.. *PROGRAM TERMINATED--FAILED-PIVOT COMPUTATION**')
   RETURN ERR1
   ENTRY PERM

   SUBROUTINE PERM(UTE) COMPUTES THE NEW TABLEAU
   GIVEN A VALUE OF THE ENTERING VARIABLE'S COLUMN
   (NEVC) AND THE DEPARTING VARIABLE'S ROW(NDVR).

```



```

C      SWAP HEADINGS FOR ROW NDVR AND COLUMN NEVC
C
DO 11 I=1,2
  J=JCOL(NEVC,I)
  JCOL(NEVC,I)=JROW(NDVR,I)
11 JFOW(NDVR,I)=J
C
C      SWAP VECTORS FOR TL(NDVR,NP) AND(TT(NP,NEVC),NP=1,NPRI)
C
DO 12 NP=1,NPRI
  TEMP=TL(NDVR,NP)
  TL(NDVR,NP)=TT(NP,NEVC)
12 TT(NP,NEVC)=TEMP
C
C      COMPUTE NEW TE ARRAY
C      PIV IS THE PIVOT ELEMENT OF THE TE MATRIX.
C
PIV=TE(NDVR,NEVC)
PIB=TB(NDVR)
DO 14 NO=1,NCBJ
  IF(NO.EQ.NDVR)GO TO 14
  PIX=TE(NO,NEVC)/PIV
  TR(NO)=FIX(TB(NO)-PIX*PIB)
DO 13 NC=1,NCCL
  IF(NC.EQ.NEVC)GO TO 13
  TE(NO,NC)=FIX(TE(NO,NC)-TE(NDVR,NC)*PIX)
13 CONTINUE
14 CONTINUE
C
C      CHANGE THE ENTRIES OF THE NDVR ROW THUS--
C
DO 15 NC=1,NCCL
  TE(NDVR,NC)=FIX(TE(NDVR,NC)/PIV)
15 TE(NDVR,NC)=FIX(TE(NDVR,NC)/PIV)
C
C      CHANGE THE ENTRIES OF THE NEVC COLUMN THUS--
C

```

```

DO 16 NO=1,NOBJ
16 TE(NO,NEVC)=FIX(-TE(NO,NEVC)/PIV)
   TB(NDVR)=FIX(TB(NDVR)/PIV)
   TE(NDVR,NEVC)=FIX(1/PIV)

C      RECOMPUTE INDEX ROWS 1 THRU NROW
C
I=1
CALL CINDX(I,NROW)
RETURN
END

```

C
C
C

```

C
C
C
C
FUNCTION FIX(Z)
      THIS FUNCTION CHANGES FLOATING POINT VALUES WITHIN
      10.**(-L) OF AN INTEGER TO THAT INTEGER.

      COMMON /PAR/ L
      Y=50./10.**L
      M=L-1
      X=.1
      DO 1 N=1,M
      X=10.*X
      I=F=X*7
      J=I-2
      DO 1 K=1,3
      G=J+K
      IF (ABS(F-G)-Y) 2,2,1
1 CONTINUE
      FIX=Z
      RETURN
2 FIX=G/X
      RETURN
      END

```

```

C C SUBROUTINE ALTST(NPFT,NROW)
C C
C C SUBROUTINE ALTST (ALTERNATE SOLUTION TEST) CHECKS
C C THE SOLVED TABLEAU FOR ALTERNATE VALID SOLUTIONS.
C C IF ONE OR MORE VALID SOLUTIONS EXISTS, ALTST
C C GENERATES AND OUTPUTS THEM.
C C
COMMON TL(105,5), TT(6,220), TE(105,220), TI(6,220),
1 TB(105), TA(6), JCOL(220,2), JROW(105,2),
2 X1(114,2), NOBJ,NPRI,NVAR,NCOL
DO 5 NC=1,NCOL
C C THE TI FIELD IS TESTED FOR AN ALL-ZERO COLUMN,
C C INDICATING ALTERNATE SOLUTIONS ON THAT COLUMN.
C C
DO 1 NP=1,NPRI
IF(TI(NP,NC).NE.0) GO TO 5
1 CONTINUE
DO 4 NO=1,NOBJ
IF(TE(NO,NC).LE.0..OR.TB(NO)..E.0.) GO TO 4
C C IF A + VALUE OF TE IS FOUND AND TB CORRESPONDING TO
C C IT IS POSITIVE, PERM IS CALLED TO PIVOT ON THAT TE
C C ELEMENT.
C C
CALL PERM(NC,NO,NROW)
DO 2 NB=1,NOBJ
IF(TB(NB).LT.0) GO TO 3
C C IF THE NEW TABLEAU HAS NO NEGATIVE TB ELEMENTS, THE
C C ALTERNATE SOLUTION IS VALID.
C C
2 CONTINUE
CALL POUT(NPRT)
C C PERM IS CALLED AGAIN TO REJRN THE TABLEAU TO ITS

```

ORIGINAL FORM FOR FURTHER ALTERNATE SOLUTION SEARCH.

C
C

3 CALL PERM(NC,NO,NROW)
4 CONTINUE
5 CONTINUE
RETURN
END


```

C
C
C
C
C
SUBROUTINE POUT(NPRT)
      SUBROUTINE POUT (PRINT OUT) FORMS AND PRINTS THE
      SOLUTION INFORMATION WITH CODED VARIABLE NAMES AND
      WITH MESSAGES INDICATING ALTERNATE SOLUTIONS.

      COMMON TL(105,5), TT(6,220), TE(105,220), TI(6,220),
1      TB(105), TA(6), JCOL(220,2), JROW(105,2),
2      X1(114,2), NOPJ, NPRI, NVAR, NCOL
      COMMON /PAR/ L
      DIMENSION WORK(114,4)
      NPRT=NPRT + 1
      IF(NPRT.NE.1) WRITE(6,31) NPRT

C
C
C
      THE OUTPUT ARRAY IS ZEROED.

      DO 12 I=1,NVAR
      DO 12 J=1,4
12  WORK(I, J)=0.

C
C
C
      THE OUTPUT ARRAY IS FILLED.

      DO 13 NP=1,NPRI
13  WORK(NP,1)=FIX(TA(NP))

C
C
C
      THE FUNCTION FIX RETURNS VALUES THAT HAVE STRAYED
      FROM BEING INTEGER VALUED DUE TO THE PRECISION
      LIMITATIONS OF THE COMPUTER, BACK TO INTEGERS.

      DO 14 NO=1,NOPJ
      IC=JROW(NO,1)
      IF=JROW(NO,2)
14  WORK(IR,IC)=FIX(TB(NO))

C
C
C
      THE OUTPUT ARRAY IS WRITTEN OUT.

```

```

WRITE(6,33) NVAR,NORJ,NOBJ,NPRI
I=MAX0(NPRI,NVAR,NORJ)
DO 20 K=1,I
  IF(K.GT.NPRI)GO TO 16
  WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4),WORK(K,1)
  GO TO 20
16 IF(K.GT.NOBJ)GO TO 17
  WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4)
  GO TO 20
17 WRITE(6,34) K,(X1(K,M),M=1,2),WORK(K,2)
20 CONTINUE
31 FORMAT(/," ALTERNATE SOLUTION NUMBER ",I3)
33 FORMAT(/," SUBSCRIPT",5X,"VARIABLE CODE",8X,I6," TERMS XSTAR",
1 1X,I6," TERMS PSTAR",1X,I6," TERMS NSTAR",2X,I6," TERMS ZBAR")
34 FORMAT(" ",I6,5X,2A10,4F19.8)
RETURN
ENTRY ERRORS
WRITE(6,35) NPRT
35 FORMAT("//" **ALGORITHM DID NOT FINISH AFTER ",I3," ITERATIONS**")
RETURN
END

```



```

10 WRITE 28
    RETURN
28 FORMAT(1H "**INPUT VARIABLE EXCEEDS ALLOWABLE DIMENSION RANGE**")
35 FORMAT(" ",(8(2X,F13.6)))
36 FORMAT(1H1///" PROBLEM READ IN SUCCESSFULLY")
37 FORMAT(" THERE WERE ",I3," BLOCKS OF 1E AND 1B CREATED"//
1 " THE MAXIMUM NUMBER OF ROWS IN A BLOCK IS ",I3)
    END

```

SUBROUTINE COEF

THIS SUBROUTINE DEVELOPS THE NONZERO COEFFICIENTS OF THE
INITIAL NONBASIC VARIABLES OF THE TE MATRIX AND INSERTS
THE CORRECT VALUES INTO THE TB MATRIX.

COMMON TE(60,380),TB(344),NORJ,NVAR,NBLK,INDEX

THESE ARRAYS SERVE AS WORKING STORAGE.
THE DIMENSIONS OF ALFA MUST BE AT LEAST (JOB CAT,
JTIMES) AND OF THE OTHER ARRAYS AT LEAST JTIMES.
DELI CONTAINS INTEGER VALUES, THE NUMBER OF
TIME PERIODS TO COMPLETE TECH TRAINING: ITS
DIMENSION IS AT LEAST JOB CAT.

DIMENSION ALFA(10,20),TH0(20),WORK1(20),WORK2(20)
INTEGER DELI(10)
N=NBK=0

THERE WILL BE JOB CAT JOB CATEGORIES AND JTIMES TIME PERIODS.
SWITCH IS A SWITCH FOR CONSTRAINT SET II. IF SWITCH
=0, CERTAIN CODE IN THAT SET IS AVOIDED. OTHERWISE,
TE COEFFICIENTS AND TB VALUES FOR THAT SET CHANGE
IN SIGN.

READ(5,*) JOB CAT,JTIMES,SWITCH
NEW=JOB CAT*JTIMES
READ(5,*) (TH0(K),K=1,JOB CAT),((ALFA(I,J),I=1,JOB CAT),J=1,JTIMES)

ALFA(I,J) IS THE ATTRITION RATE FROM THE THEATER
FOR JOB I IN TIME (J-1,J).
DETERMINE THE SURVIVAL RATE ALFA(I,J)=1-ALFA(I,J)
TH0(K) IS THE INITIAL LEVEL IN THE CONUS FORCE FOR
JOB CATEGORY K.

```

DO 11 J=1,JTIMES
DO 11 I=1,JORCAT
11 ALFA(I,J)=1.-ALFA(I,J)
DO 12 I=1,JORCAT
12 TR(I)=TB(I)-THO(I)*ALFA(I,1)

CONSTRAINT TYPE I: OVERSEAS AVAILABILITY

GENERAL EQUATIONS:

(1-ALFA(I,J))*THETA(I,J-1)+X(I,J)-C(I,J)+N-P=RBAR(I,J)
FOR I=1,2,...,JORCAT FOR J=2,3,...,JTICES
X(I,1)-C(I,1)+N-P=RBAR(I,1)-THO(I)*(1-ALFA(I,1))
FOR I=1,2,...,JORCAT

N IS THE NEGATIVE DEVIATION FROM THE GOAL:
P IS THE POSITIVE DEVIATION FROM THE GOAL:
X(I,J) IS A FLOW FROM CONUS TO THE THEATER IN TIME
(J-1,J) FOR JOB CATEGORY I
C(I,J) IS A FLOW FROM THE THEATER TO CONUS IN TIME
(J-1,J) FOR JOB CATEGORY I
RBAR(I,J) IS AN INITIAL TB VALUE (DESIRED RESIDUAL
FORCE SIZE)

CALL BEGIN(NEW)

CALL BEGIN(K) MAKES THE FIRST K ROWS OF TE EQUAL
TO ZERO.

DO 14 J=1,NEW
J2=MOD(J,JORCAT)

THIS STRUCTURE OF TWO DO _OOPS WITH THE SAME
TERMINAL STATEMENT PREPARES THE TE COEFFICIENT
IN THEIR LOWER TRIANGULAR FORM FOR THE X'S AND C'S.
J=1,NEW IMPLIES THE OUTER DO LOOP WORKS ON

```

```

C C C C C
      ONE COLUMN AT A TIME.
      I=J,NEW,JORCAT IMPLIES THE INNER DO LOOP
      WORKS ON THE ROWS FROM ROW J TO ROW NEW
      IN INCREMENTS OF JORCAT.

      IF(J2.EQ.0) J2=JORCAT
      DO 14 I=J,NEW,JORCAT
      IF(I.EQ.J) GO TO 13
      I2=(I-1)/JORCAT+1

      J2 AND I2 ARE CREATED TO GET THE PROPER COEFFICIENTS
      OF ALFA.

      TE(I,J)=TE(I-JORCAT,J)*ALFA(J2,I2)
      TE(I,J+NEW)=-TE(I,J)
      IF(I.EQ.J+JORCAT) TR(I)=TB(I)-TH0(J2)*ALFA(J2,1)*TE(I,J2)
      GO TO 14
13  TE(I,J)=1.
14  TE(I,J+NEW)=-1.
      CONTINUE

      CALL FINISH(K,N) CREATES THE RECORDS OF THE
      CONSTRAINT SET FOR TAPE1 AND DIRECTS THE WRITING
      TO THE FILE. N IS RETURNED AS N+K.

      CALL FINISH(NEW,N)

      CONSTRAINT TYPE II: CONUS AVAILABILITY

      GENERAL EQUATIONS:

      K(I,J-1)+C(I,J)-X(I,J)-V(I,J)+Y(I,J)+(1-LAMBDA(I))*W(I,J-DELI(I))
      +U(I,J) + N - P = K(I,J) I=1,2,...,JORCAT:
      J=2,3,...,JTIMES
      C(I,1)-X(I,1)-V(I,1)+Y(I,1)+(1-LAMBDA(I))*W(I,1-DELI(I))
      +U(I,1) + N - P = K(I,1)-K(I,0) I=1,2,...,JORCAT

```

```

C C C V(I,J) IS THE FLOW RATE FROM CONUS TO INSTRUCTORS IN
C C C JOB I FOR TIME (J-1,J)
C C C U(I,J) IS THE FLOW RATE FROM INSTRUCTORS TO CONUS IN JOB I
C C C FOR TIME (J-1,J)
C C C Y(I,J) IS THE FLOW RATE OF RESERVES TO CONUS FOR JOB I
C C C IN TIME (J-1,J)
C C C W(I,J) IS THE FLOW RATE OF JOB I FROM BASIC TRAINING TO
C C C TECH TRNG IN TIME (J-1,J)
C C C LAMBDA(I) IS THE ATTRITION RATE OF JOB FOR TECH TRNG
C C C K(I,J) IS THE LEVEL OF JOB I IN CONUS AT TIME J.
C C C READ(5,*)(TH0(K),DELI(K),K=1,JOB CAT)
C C C
C C C ALL OF THE LAMBDA VALUES ARE IN TH0.
C C C CREATE SURVIVAL RATES IN I+0.
C C C
C C C DO 15 K=1,JOBCAT
C C C 15 TH0(K)=1.-TH0(K)
C C C NEW2=NEW+NEW
C C C CALL BEGIN(NEW)
C C C DO 18 J=1,NEW
C C C J2=J+NEW
C C C NEXTC=NEW*5+J
C C C II=0
C C C DO 18 I=J,NEW,JOBCAT
C C C IF(II.GT.0) GO TO 16
C C C I2=MOD(I,JOBCAT)
C C C IF(I2.EQ.0) I2=JOBCAT
C C C 16 II=II+1
C C C IF(II.LE.DELI(I2)) GO TO 17
C C C
C C C UNTIL II IS GREATER THAN THE NUMBER OF TIME PERIODS
C C C TO FINISH TECH TRNG, AVOID THE NEXT LINE OF CODE.
C C C TE(I,NEXTC)=TH0(I2)

```



```

C
C
C
C
17 TE(I,J)=TE(I,J+NEW2)=-1.
18 TE(I,J2)=TE(I,J2+NEW2)=TE(I,J+NEW2*2)=1.
C
C
C
C
    FOR A NONZERO SWITCH, CHANGE THE SIGN OF THE
    TB VALUES AND TE COEFFICIENTS FOR THIS CONSTRAINT.
C
C
C
C
    IF(SWITCH.EQ.0.) GO TO 20
    DO 19 J=1,NVAR
    DO 19 I=1,NEW
    19 TE(I,J)=-TE(I,J)
    20 CALL FINISH(NEW,N)
C
C
C
C
    WORK1 HOLDS THE INITIAL K(I) VALUES AND ALFA HOLDS THE
    INITIAL W(I) VALUES.
C
C
C
C
    DO 21 K=1,JORCAT
    K1=DELI(K)
    READ(5,*)(WORK1(K),(ALFA(K,J),J=1,K1))
C
C
C
C
    MAKE CUMULATIVE SUMS IN ALFA.
C
C
C
C
    DO 21 J=2,K1
    21 ALFA(K,J)=ALFA(K,J)+ALFA(K,J-1)
    J=DELI(1)
C
C
C
C
    FIND THE MAXIMUM NUMBER OF PERIODS TO COMPLETE TECH
    TRNG (MAX OVER ALL JOB CATEGORIES) AND PUT THIS
    VALUE IN J.
C
C
C
C
    DO 22 I=2,JORCAT
    IF(J.LT.DELI(I))J=DELI(I)
    22 CONTINUE
C
C
C
C
    MAKE DUPLICATE COPIES OF ALFA(I,DELI(I)) IN
    ALFA(I,K) FOR K=DELI(I) TO K=J. EVENTUALLY
    THE TB VALUES ARE PROPERLY TRANSLATED.

```

C

```

DO 23 I=1, JOBCAT
I3=DELI(I)
DO 23 K=I3, J
23 ALFA(I,K)=ALFA(I,I3)
DO 24 I=1, NEW
I2=MOD(I, JOBCAT)
IF (I2.EQ.0) I2=JOBCAT
I3=(I-1)/JOBCAT+1
IF (I3.GT.J) I3=J
24 TB(NEW+I)=TB(NEW+I)-WORK1(I2)-TH0(I2)*ALFA(I2,I3)
IF (SWITCH.EQ.0.) GO TO 26
K=NEW+1
DO 25 I=K, NEW2
25 TB(I)=-TB(I)

```

CONSTRAINT TYPES III AND IV: MOVEMENT FROM THE RESERVES

GENERAL EQUATIONS:

```

Y(I,1) + Y(I,2) + ... + Y(I,JTIMES) + N - P = R(I,0)-RBAR(I)
FOR I=1,2,...,JOBCAT
Y(I,J) + N - P = YBAR(I,J)
FOR I=1,2,...,JOBCAT AND J=1,2,...,JTIMES

```

R(I,0) IS THE INITIAL RESERVE FORCE LEVEL
 RBAR(I) IS A DESIRED RESIDUAL FORCE SIZE OF THE RESERVES
 YBAR(I,J) IS AN ACTIVATION LIMIT ON THE RESERVES.

```

26 NEW3=NEW+NEW2
CALL REGIN(JOBCAT)
DO 27 I=1, JOBCAT
DO 27 J=1, NEW, JOBCAT
27 TE(I,NEW3+J)=1.
CALL FINISH(JOBCAT,N)
CALL BEGIN(NEW)

```

C C C C C C C C C C C C C C C C

```

DO 28 I=1,NEW
28 TE(I,NEW3+I)=1.
CALL FINISH(NEW,N)

CONSTRAINT TYPE V: INPUTS INTO TECHNICAL TRAINING

GENERAL EQUATIONS:


$$W(1,J) + W(2,J) + \dots + W(JOBCAT,J) + BETA * E(J-1-LAG) + N - P = 0$$

FOR EVERY J=1,2,...,JTIMES

NEXTC IS THE NEXT C(COLUMN) TO CONSIDER.

NEXTC=NEW3+NEW2
NEW6=NEW3+NEW3

READ LAG, BETA, WORK1(I) FOR I EQUAL 1 TO LAG + 1.
LAG IS THE TIME TO FINISH BASIC TRAINING (AN INTEGER
NUMBER OF PERIODS). BETA IS THE SURVIVAL FACTOR OF
THAT TRAINING. WORK1(I) HOLDS THE INITIAL VALUES
OF E (THAT IS, E(-1) AND E(0) FOR A LAG OF 1),
RESPECTIVELY. E REFERS TO THE INPUT INTO BASIC
TRAINING.

READ(5,*) LAG,BETA
LAG1=LAG+1
READ(5,*) (WORK1(I),I=1,LAG1)
CALL BEGIN(JTIMES)
DO 30 J=1,JTIMES
DO 29 I=1,JOBCAT
29 TE(J,NEXTC+I)=1.
IF(J.GT.LAG1) TE(J,NEW6+J-LAG1)=-BETA
IF(J.LE.LAG1) TR(N+J)=WORK1(J)*BETA
30 NEXTC=NEXTC+JOBCAT
CALL FINISH(JTIMES,N)

```



```

I2=MOD(I,JOB CAT)
IF(I2.EQ.0) I2=JOB CAT
TF(I,NEXTC)=PERCT
TF(I,NEXTC-NEW2)=-PERCT
IF(I.EQ.J+JOB CAT) TR(N+I)=PERCT*WORK2(I2)
GO TO 35
34 TE(I,NEXTC)=1.
IF(J.LE.JOB CAT) TB(N+J)=PERCT*(WORK1(J)+ALFA(J,1)-ALFA(J,2))
35 CONTINUE
CALL FINISH(NEW,N)
N=0

CALL RHSTE(K,N) CREATES THE RIGHT HAND SIDE OF THE
TE MATRIX. THE RIGHT HAND SIDE HAS NOBJ COLUMNS,
AND THE LEFT HAND SIDE HAS NVAR COLUMNS.
RHSTE WORKS SIMILARLY TO RJJTINE FINISH IN WRITING
BLOCKS OF ROWS ON TAPE1.

CALL RHSTE(NEW,N)
CALL RHSTE(NEW,N)
CALL RHSTE(JOB CAT,N)
CALL RHSTE(NEW,N)
CALL RHSTE(JTIMES,N)
CALL RHSTE(NEW,N)
CALL RHSTE(JTIMES,N)
CALL RHSTE(NEW,N)
RETURN
END

```

C C C C C C C

```

C
C
C
SUBROUTINE BEGIN(NN)
      THIS ROUTINE INITIALIZES THE NN ROWS OF TE TO ZERO.
      COMMON TE(60,360),TB(344),NOBJ,NVAR,NBLK,INDEX
      DO 20 NOR=1,NVAR
      DO 20 NO=1,NN
20  TE(NO,NOR)=0.
      RETURN
      END

```


C

```
NBLK=NBLK+1
N1=N1-INDEX
IF(N1.GT.0) GO TO 2F
N=N+NN
RETURN
END
```



```

PROGRAM MAIN2(INPUT=3008/80,OUTPJT=30008,TAPE1=240009,TAPE3=09,
1 TAPE2=240008,TAPE4=08,TAPE5=INPJT,TAPE6=OUTPUT,TAPE8=167008)

```

```

THE COMPUTATIONAL TABLEAU IS BROKEN INTO FIVE ARRAYS:

```

```

TTE(NO,NC) = EQUATION FIELD
TR(NO) = RIGHT HAND SIDE 3DA-S
TT(NP,NC) = TOP STUB AND INDEX ROWS
TL(NO,NP) = LEFT STUB VALUES
TA(NP) = GOAL ACHIEVEMENT VECTOR

```

```

TT IS A STORAGE ARRAY FOR TOP STUB AND INDEX VALUES: ONE
OR THE OTHER (AND NOT BOTH) IS NEEDED IN MEMORY AT ONE
TIME. ONLY IN SUBROUTINE CINDX DO BOTH SETS INTERACT: HOWEVER,
INDEX ROWS ARE COMPUTED DIRECTLY FROM TOP STUB VALUES. AFTER
THIS COMPUTATION, THE TOP STUB VALUES ARE NOT NEEDED AGAIN
UNTIL CALLED FOR BY POUTINE PERM OR CINDX.

```

```

THE SUBSCRIPTS ABOVE INDICATE:

```

```

NO=1,NORJ THE NUMBER OF OBJECTIVE EQUATIONS
NC=1,NCOL THE NUMBER OF DECISION VARIABLES PLUS THE
NUMBER OF OBJECTIVE EQUATIONS
(FOR TTE MATRIX, NC=1, MAX(NORJ,NVAR) )
NP=1,NPRI THE NUMBER OF PRIORITIES

```

```

THE COLUMN AND ROW HEADINGS FOR THE FINAL PRINTOUT ARE
ENCODED INTO THE JCOL(NSUR,NTYPE) AND JROW(NSUR,NTYPE)
FIELDS.

```

```

NSUB = 2 IMPLIES A DECISION VARIABLE X
      = 3 IMPLIES A POS. DEVIATION P
      = 4 IMPLIES A NEG. DEVIATION N

```

```

ARRAY IN(8,3) MAINTAINS THE INFORMATION ON HOW TAPE1 IS
INITIALLY CREATED. FOR EACH CONSTRAINT SET I,

```

```

IN(I,1) IS THE NUMBER OF EQUATIONS IN THE SET.

```

```

IN(I,2) IS THE REMAINDER OF IN(I,1)/INDEX

```

```

IN(I,3) IS THE NUMBER OF BLOCKS CREATED FOR THE SET.

```

```

C C SCALARS IN THE COMMON AREA ARE
C C NOBJ = NUMBER OF OBJECTIVE EQUATIONS
C C NPRI = NUMBER OF PRIORITY LEVELS
C C NVAR = NUMBER OF DECISION VARIABLES
C C NCOL = NOBJ+NVAR
C C NBLK = TOTAL NUMBER OF BLOCKS ON TAPE1
C C NTT = SWITCH FOR INDICATING TOP STUF OR INDEX
C C
C C ROWS (1 FOR TOP STUF AND 2 FOR INDEX ROWS)
C C NTAPE = SWITCH FOR INDICATING FILE OF CURRENT TTE
C C
C C VALUES (1 FOR TAPE1 AND 2 FOR TAPE2)
C C INDEX = MAXIMUM NUMBER OF ROWS PER BLOCK ON TAPE1.
C C NROW1 = THE LAST PRIORITY LEVEL. WORKED ON BY THE PROGRAM
C C PRIOR TO THE CALL TO SJROUTINE STORE.
C C
C C COMMON TTE(20,390),IT(6,724),IB(344),IL(344,6),JCOL(724,2),
1 JROW(344,2),IN(8,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2 NTT,NTAPE,INDEX,NROW1
C C
C C L IS A POSITIVE INTEGER GREATER THAN 1 WHICH
C C SPECIFIES THE NUMBER OF DECIMAL PLACES OF PRE-
C C CISION USED IN FUNCTION FIX.
C C
C C IMAX IS THE MAXIMUM NUMBER OF TABLEAU GENERATIONS
C C ALLOWED BEFORE A RECURSIVE LOOP IS ASSUMED.
C C NPRT IS THE NUMBER OF PRINTOUTS PROCESSED.
C C
C C COMMON /PAR/ L
C C DATA IMAX,NPRT/400,0/
C C T1=SECOND(T)
C C REWIND 1
C C NTAPE=1
C C
C C SUBROUTINE INIT2 INITIALIZES THE INPUT FILES
C C ACCORDINGLY. FAULTY OR MISPLUNCHED DATA CAN CAUSE
C C ALTERNATE RETURNS TO STATEMENT 5.

```

```

C      CALL INIT2, RETURNS(5)
C
C      WHEN NROW BECOMES NPRI+1, THE ALGORITHM FOR DETER-
C      MINING A FINAL SOLUTION IS ENDED. NROW IS THE
C      ROW THAT THE PROGRAM IS CURRENTLY USING IN THE
C      MULTIPHASE OPTIMIZING ROUTINE.
C
C      DO 4 NROW=1, NPRI
C      PRINT *, "THE PRIORITY LEVEL IS ", NROW
C
C      CALLING CINDX(NROW) CAUSES THE INDEX ROWS 1,2,...,NROW
C      TO BE COMPUTED.
C
C      CALL CINDX(NROW)
C
C      SUBROUTINE TEST COMPUTES THE ENTERING VARIABLE
C      COLUMN (NEVC) AND THE DEPARTING VARIABLE ROW (NDVR).
C      WHEN NEVC=0, THE ROW CANNOT BE OPTIMIZED FURTHER.
C
C      CALL TEST(NEVC, NDVR, NROW), RETURNS(5)
C      DO 3 IVAL=1, IMAX
C      PRINT *, "IVAL, NDVR, AND NEVC ARE ", IVAL, " ", NDVR, " ", NEVC
C      IF(NEVC.LE.0) GO TO 4
C
C      SUBROUTINE PERM COMPUTES THE NEW TABLEAU.
C
C      CALL PERM(NEVC, NDVR, NROW)
C      IF(SECOND(T)-T1.LE.6000.0) GO TO 3
C      CALL POUT(NPRT)
C      CALL STORE(NROW)
C      3 CALL TEST(NEVC, NDVR, NROW), RETURNS(5)
C
C      THE DO 3 LOOP ENDS BY REACHING ITS UPPER LIMIT
C      WITHOUT CONVERGING.
C

```

```

CALL ERRORS(IMAX)
GO TO 5
4 CONTINUE
  NROW=NPRI
  SUBROUTINE POUT IS CALLED TO PRINT OUT THE DESIRED
    SOLUTION.
  CALL POUT(NPRT)
  SUBROUTINE ALTST TESTS FOR ALTERNATIVE VALID SOLUTIONS
    AND PRINTS OUT ANY THAT IT PRODUCES.
  CALL ALTST(NPRT,NROW)
  STOP
5 STOP "FAILURE"
  END

```

```

C
C
C
C
C
C
C
C

```

```

C C C C C C
SUBROUTINE INIT2, RETURNS (ERR1)

      THIS ROUTINE ESTABLISHES THE INITIALIZATION OF THE
      ALGORITHM BY READING IN THE INPUT DATA AND USING THIS
      DATA TO CALCULATE THE INITIAL COEFFICIENTS FOR THE
      APPROPRIATE MATRICES.

      COMMON TTE(20,380), TT(6,724), FB(344), IL(344,6), JCOL(724,2),
1     JROW(344,2), IN(8,3), TA(5), NOBJ, NPRI, NVAR, NCOL, NBLK,
2     NTT, NTAPE, INDEX, NROW1

      COMMON /PAR/ L
      COMMON /LABELS/ X1(380,2)
      DIMENSION IPRI(4), ISUB(4), WHT=(4)

      THE INPUT DATA IS READ IN THE FOLLOWING MANNER:

      (LIST-DIRECTED)
      FIRST CARD  NOBJ, NPRI, NVAR, NBLK, NUMBER OF TERMS
                  OF THE ACHIEVEMENT FUNCTION, L, THE NUMBER
                  OF JOB CATEGORIES, THE NUMBER OF TIME PERIODS,
                  INDEX

      SECOND GROUP OF CARDS
                  OF THE FORM: I, IPRI, ISUB, WHTF
                  WHERE I IS 4 FOR EVERY CARD EXCEPT FOR THE FINAL CARD
                  WHERE I MAY BE 1, 2, 3, OR 4. IPRI IS THE PRIORITY LEVEL,
                  ISUB IS THE DEVIATION VARIABLE, AND WHTF IS THE WEIGHTING
                  FACTOR OF THAT DEVIATION VARIABLE AT THAT PRIORITY LEVEL.

      (FORMATTED--9A10)
      FINAL GROUP OF CARDS
                  THE LABELS OF EACH DECISION VARIABLE, EACH LABEL OCCUPY-
                  ING NO MORE THAN 20 CHARACTERS (MAX OF FOUR LABELS PER
                  CARD).

      READ(5,*) NOBJ, NPRI, NVAR, NBLK, NTAPE, L, JOBCAT, JTIMES, INDEX

```



```

C
C
C
C
      IF(EOF(5).NE.0) RETURN ERR1
      INPUTS ARE CHECKED TO ENSURE THAT THEY DO NOT
      EXCEED THE MAXIMUM DIMENSIONS OF THE ARRAYS.

      IF(NOBJ.LT.1.OR.NOBJ.GT.343) GO TO 10
      IF(NPRI.LT.1.OR.NPRI.GT.6) GO TO 10
      IF(NVAR.LT.1.OR.NVAR.GT.380) GO TO 10
      NCOL=NOBJ+NVAR

C
C
C
      COLUMN AND ROW HEADINGS ARE SET.

      DO 1 NV=1,NVAR
      JCOL(NV,1)=2
1     JCOL(NV,2)=NV
      DO 2 NO=1,NOBJ
      NC=NO+NVAR
      JCOL(NC,1)=3
      JROW(NC,1)=4
2     JCOL(NC,2)=JROW(NC,2)=NO
      DO 4 NP=1,NPRI
      DO 3 NO=1,NOBJ
3     TL(NO,NP)=0.
      DO 4 NC=1,NCOL
4     TT(NP,NC)=0.

C
C
C
C
      NO IS THE NUMBER OF CARDS CONTAINING ACHIEVEMENT
      FUNCTION VALUES.

      NC=NTAF/4
      IF(MOD(NTAF,4).NE.0) NO=NO+1
      DO 5 NT=1,NO
      READ(5,*) KK,(IPRI(K),ISUB(K),HTF(K),X=1,KK)
      DO 5 K1=1,KK

C
C
      SUBROUTINE PLACE INSERTS VALUES INTO STUBS.

```

```

C C C C C C C
      KK IS THE NUMBER OF COMBINATIONS TO BE READ ON THIS
      NEXT INPUT CARD. ONLY ON THE FINAL CARD OF THESE
      VALUES CAN THERE BE FOUR OR FEWER COMBINATIONS.

      5 CALL PLACE(IPRI(K1),ISUB(K1),JTF(K1)),RETURNS(30)

      GENERATE THE VALUES OF THE IN MATRIX.

      DO 6 K1=1,9
      IF(K1.EQ.3) IN(K1,1)=JOB CAT
      IF(K1.EQ.5.OR.K1.EQ.7) IN(K1,1)=JTIMES
      IF(K1.NE.3.AND.K1.NE.5.AND.K1.NE.7) IN(K1,1)=JOB CAT*JTIMES
      IN(K1,2)=MOD(IN(K1,1),INDEX)
      IN(K1,3)=IN(K1,1)/INDEX
      IF(IN(K1,2).NE.0) IN(K1,3)=IN(K1,3)+1
      6 CONTINUE
      NB=NB/K-1

C C C C C
      THE LAST RECORD OF TAPE1 IS THE COLLECTION OF
      RIGHT HAND SIDE GOAL COEFFICIENTS. THE PROGRAM
      READS IN THESE VALUES INTO THE TB MATRIX.

      DO 7 NA=1,NB
      7 READ (NTAPE)
      READ (NTAPE) (TB(NA),NA=1,NORJ)

C C C C C C C
      TAPE3 WILL HOLD THE TOP STJ3 VALUES AS RECORD 1
      AND INDEX ROWS AS RECORD 2. ONCE SUBROUTINE
      PLACE HAS CREATED THE TOP STJB VALUES, THE PROGRAM
      WRITES THEM TO THE FILE. JTF BECOMES 1.

      READ THE LABELS NOW FOR THE DECISION VARIABLES FROM
      TAPE5.

      READ(5,15) ((X1(I,J),J=1,2),I=1,NVAR)

```

```

RFINDD 3
BUFFER OUT (3,1) (TT(1,1),TT(NPRI,NCOL))
IF (UNIT(3)) 8,30,30
8 NTT=1
RETURN
10 WRITE 20
20 FORMAT(1H " **INPUT VARIABLE EXCEEDS ALLOWABLE DIMENSION RANGE**")
15 FORMAT((8A10))
30 RETURN ERR1
END

```

```

C C C C C C C C C C
SUBROUTINE PLACE(JPRI,JSUB,WHT),RETURNS(ERR1)

SUBROUTINE PLACE POSITIONS THE ACHIEVEMENT FUNCTION
VALUES IN LEFT HAND AND TOP STUBS.

      JPRI      PRIORITY LEVEL
      JSUB      +JSUB = P SUB JSJ3
      WHT       -JSUB = N SUB JSJ3
                WEIGHTING FACTOR

      COMMON TTE(20,380),IT(6,724),IB(344),TL(344,6),JCOL(724,2),
1      JROW(344,2),IN(8,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2      NTT,NTAPE,INDEX,NROW1

      INPUT DATA IS CHECKED TO BE CERTAIN THAT IT IS IN
      RANGE.

      IF(JPRI.LT.1.OR.JPRI.GT.NPRI.OR.JSUB.EQ.0.OR.IABS(JSUB).GT.NOBJ)
1 GO TO 2
C C C C C C C C C C
      POSITIVE VALUES OF JSUB INDICATE TOP STUB VALUES.

      IF(JSUB.GT.0) GO TO 1
C C C C C C C C C C
      PLACE VALUE IN LEFT STUB

      JSUB= -JSUB
C C C C C C C C C C
      OVERWRITES ARE DETECTED AS BEING AN ERROR.

      IF(TL(JSUB,JPRI).NE.0) GO TO 3
      TL(JSUB,JPRI) = WHT
      RETURN
1 ICOL= JSUB + NVAR
C C C C C C C C C C
      PLACE VALUE IN RIGHT STUB

```

C

```
IF(TT(JPRI,ICOL).NE.0) GO TO 3
TT(JPRI,ICOL) = WHT
RETURN
2 WRITE(6,4)
GO TO 6
3 WRITE(6,5)
4 FORMAT(//, "***SUBSCRIPT OUT-OF-RANGE WHILE READING SUBSCRIPTS**")
5 FORMAT(//, "***OVERWRITE ATTEMPTED IN A STUB**")
6 RETURN ERR1
END
```


SUBROUTINE CINDX(NROW)

THE FUNCTION OF CINDX IS TO COMPUTE INDEX ROWS.
CINDX DOES THIS FROM ROW 1 TO ROW NROW.
THE TOP STJB VALUES MUST BE IN MEMORY. HENCE,
CHECK TO SEE IF NTT=1. IF NOT, THE PROGRAM READS
IN RECORD 1 OF TAPE3 AND SETS NTT=1.

```
COMMON TTE(26,330),TT(6,724),TB(344),TL(344,6),JCOL(724,2),
1     JROW(344,2),IN(8,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2     NTT,NTAPE,INDEX,NROW1
3     IF(NTT.EQ.1) GO TO 2
4     REWIND 3
5     BUFFER IN (3,1) (TT(1,1),TT(NPRI,NCOL))
6     IF(UNIT(3)) 1,11,11
7     NTT=1
8     DO 9 NP=1,NROW
9     TA(NP)=0.
```

THE RIGHT HAND SIDE OF THE INDEX ROWS ARE COMPUTED.

```
DO 3 NO=1,NOBJ
3 TA(NP)=TA(NP)+TB(NO)*TL(NO,NP)
DO 4 NC=1,NCOL
4 TT(NP,NC)=-TT(NP,NC)
REWIND NTAPE
```

CINDX NEEDS EVERY ELEMENT OF THE TTE MATRIX FOR
COMPUTING INDEX ROWS. THE TAPE (NTAPE) IS READ
SEQUENTIALLY--FIRST, THE ORIGINAL LEFT HAND
SIDE AND SECOND, THE ORIGINAL RIGHT HAND SIDE.

NO IS A COUNTER RANGING FROM 1 TO NOBJ. IT IS
INITIALLY SET TO 0 AND SPECIFIES THE NUMBER OF
COLUMNS TO BE READ IN. FOR LEFT HAND TTE VALUES,
N=NVAR: FOR RIGHT HAND VALUES, N=NOBJ.


```

C C SUBROUTINE TEST(NEVC,NDVR,NROW), RETURNS(ERR1)
C C
C C SUBROUTINE TEST COMPUTES THE NEXT ENTERING
C C VARIABLE'S COLUMN AND DEPARTING VARIABLE'S ROW. IF
C C THE SEARCH OF INDEX ROW INDICATES THAT NO FURTHER
C C OPTIMIZATION MAY TAKE PLACE ON LEVEL NROW, A FLAG
C C VALUE OF NEVC=0 IS RETURNED. OTHERWISE, NEVC IS
C C RETURNED AS THE INDEX COLUMN HAVING THE LARGEST
C C POSITIVE VALUE WITH NO NEGATIVE VALUES IN A HIGHER
C C INDEX. TIES ARE BROKEN BY RIGHTMOST OPTIMIZATION.
C C THE ENTERING ROW IS CHOSEN AS HAVING THE SMALLEST
C C POSITIVE RATIO OF TB(NEVR)/TE(NEVR,NEVC) FOR
C C POSITIVE ELEMENTS ONLY.
C C TIES ARE BROKEN BY HIGHEST PRIORITY DOMINATION:
C C FURTHER TIES ARE BROKEN BY LOWEST ROW DOMINATION.
C C
C C COMMON TTE(20,330), TT(5,724), TB(344), TL(344,6), JCOL(724,2),
1 JROW(344,2), IN(8,3), TA(5), NOBJ, NPRI, NVAR, NCOL, NBLK,
2 NIT, NTAPE, INDEX, NROW1
C C COMMON /PAR/ L
C C
C C T2 IS THE ARRAY OF VALUES FROM THE ENTERING VARIABLE
C C COLUMN. T1 IS THE ARRAY OF VALUES FROM THE
C C DEPARTING VARIABLE ROW. T1 IS USED ONLY IN PERM
C C WHICH IS THE SECOND ENTRY POINT OF THE ROUTINE.
C C
C C COMMON /DATA1/ T2(344)
C C DIMENSION T1(724)
C C NEVC=0
C C
C C IF THE LEVEL OF ACHIEVEMENT OF PRIORITY NROW IS
C C ZERO, THIS LEVEL IS OPTIMUM--RETURN NEVC = 0
C C
C C IF(TA(NROW).LE.0.) RETURN
C C
C C DETERMINE COLUMN OF ENTERING VARIABLE

```


C C FILE AND READ FROM THE MIDDLE OF THE FILE ONWARD.

```

NT=NBLK/2
N=NOBJ
DO 4 NT1=1,NT
4 READ (NTAPE)
5 DO 11 I=1,8
K2=IN(I,2)
K3=INDEX
J=IN(I,3)
DO 10 LL=1,J

```

C C ONLY FOR THE FINAL BLOCK IF A CONSTRAINT SET IS THE
C C NUMBER OF ROWS EQUAL TO K2=IN(I,2).

```

IF(LL.EQ.J.AND.K2.NE.0) K3=K2
READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
DO 9 K=1,K3
NO=NO+1
T2(NO)=TTE(K,NE)
IF(T2(NO).LE.0) GO TO 9
V=TP(NO)/T2(NO)
IF(NDVR.EQ.0) GO TO 8
IF(V-VDVR)8,6,9
6 DO 7 NP=1,NPRI
7 IF(TL(NO,NP)-TL(NDVR,NP))9,7,8
7 CONTINUE
8 VDVR=V
NDVR=NO
9 CONTINUE
10 CCNTINUE
11 CONTINUE
IF(NDVR.GE.1) RETURN
WRITE(6,20)
20 FORMAT(" *PROGRAM TERMINATED--FAILED-PIVOT COMPUTATION**")
RETURN ERR1

```



```

C C ENTRY PERM
C C
C C SUBROUTINE PERM(UTE) COMPUTES THE NEW TABLEAU
C C GIVEN A VALUE OF THE ENTERING VARIABLE'S COLUMN
C C (NEVC) AND THE DEPARTING VARIABLE'S ROW(NDVR).
C C
C C SWAP HEADINGS FOR ROW NDVR AND COLUMN NEVC
C C
C C DO 25 I=1,2
C C J=JCOL(NEVC,I)
C C JCOL(NEVC,I)=JROW(NDVR,I)
C C 25 JFOW(NDVR,I)=J
C C
C C SWAP VECTORS FOR TL(NDVR,NP) AND TT(NP,NEVC), NP=1,NPRI)
C C ENSURE THAT TOP STUB VALUES ARE IN MEMORY AT THIS
C C TIME.
C C
C C REWIND 3
C C BUFFER IN (3,1) (TT(1,1),TT(NPRI,NCOL))
C C IF(UNIT(3)) 26,55,55
C C 26 NTT=1
C C DO 27 NP=1,NPRI
C C TEMP=TL(NDVR,NP)
C C TL(NDVR,NP)=TT(NP,NEVC)
C C TT(NP,NEVC)=TEMP
C C 27 REWIND 3
C C BUFFER OUT (3,1) (TT(1,1),TT(NPRI,NCOL))
C C
C C COMPUTE THE NEW TTE ARRAY. FIRST, FIND THE NDVR
C C VALUES AND STORE THEM IN ARRAY T1.
C C
C C IB=0
C C N=NDVR
C C
C C IB WILL EVENTUALLY CONTAIN THE BLOCK NUMBER IN
C C WHICH NDVR LIES. FOR RIGHT HAND SIDE VALUES, NDVR

```

C IS LOCATED IN THE BLOCK NJ1BER IB+NBLK/2. LL IS
 C THE NUMBER OF ROWS OF CONSTRAINT SET NT MINUS NDVR.
 C WHEN LL IS NO LONGER NEGATIVE, THE CONSTRAINT SET NT
 C HOLDS THE NDVR ROW.
 C

```

DO 30 NT=1,8
K=IN(NT,2)
LL=IN(NT,1)-N
IF(LL.GE.0) GO TO 31
N=-LL
IR=IB+IN(NT,3)
30 CONTINUE
31 IR=IB+N/INDEX
N1=MOD(N,INDEX)
IF(N1.NE.0) IB=IB+1
IF(N1.EQ.0) N1=INDEX
  
```

C J1 IS THE NUMBER OF ROWS IN BLOCK IB.
 C N1 IS THE NDVR ROW IN BLOCK IB. WITH THIS CORE-
 C SWAPPING ROUTINE, N1 IS THE CORRECT ROW COEFFICIENT
 C OF THE TTE MATRIX.
 C

```

J1=INDEX
IF(LL.LT.K) J1=K
II=IR-1
REWIND NTAPE
DO 35 NT=1,2
M=0
N=NVAR
IF(NT.NE.2) GO TO 32
M=NVAR
N=NOBJ
II=NBLK/2-1
  
```

C CHECK TO SEE IF IB=1 (OR EQUIVALENTLY II=0).
 C WE WANT TO SKIP II RECORDS ON THE TAPE. FOR II=0,
 C

```

C      WE MERELY READ THE INITIAL RECORD.
C
32 IF(II.EQ.0) GO TO 34
   DO 33 IJ=1,II
33 READ (NTAPE)
34 READ (NTAPE) ((TTE(I1,I2),I2=1,N),I1=1,J1)
   DO 35 NC=1,N
35 T1(NC+M)=TTE(N1,NC)
36 CONTINUE

C      THE PIVOT ELEMENT IS TTE(NDVR,NEVC)=T1(NEVC).
C
C      PIV=T1(NEVC)
C      PIB=TR(NDVR)
C      NN=1

C      NN REFERS TO THE SIDE OF THE TTE MATRIX WHERE NEVC
C      IS LOCATED.  WHEN NEVC EXCEEDS NVAR, NN=2.
C      OTHERWISE, NN=1.
C
NE=NEVC
IF(NEVC.GT.NVAR) NE=NE-NVAR
IF(NEVC.GT.NVAR) NN=2
REWIND NTAPE
NSCRAT=NTAPE+1

C      NSCRAT REFERS TO THE SCRATCH TAPE ON WHICH THE LATEST
C      TTE VALUES ARE WRITTEN.
C
IF(NSCRAT.GT.2) NSCRAT=1
REWIND NSCRAT
DO 44 NT=1,2
N=NVAR
M=NO=0
IF(NT.EQ.1) GO TO 37
M=NVAR

```

```

N=NOBJ
37 DO 43 I=1,8
   K2=IN(I,2)
   K3=INDEX
   J=IN(I,3)
   DO 43 LL=1, J
   IF(LL.EQ.J.AND.K2.NE.0) K3=K2
   READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
   DO 42 K=1,K3
   NC=NO+1
   IF(NC.EQ.NDVR) GO TO 40
   PIX=T2(NO)/PIV
   IF(NT.EQ.1) T3(NO)=FIX(TB(NO)-PIX*PIB)
   DO 39 NC=1,N
   IF(NC.EQ.NT.AND.NC.EQ.NE) GO TO 38
   TTE(K,NC)=FIX(TTE(K,NC)-T1(NC+1)*PIX)
   GO TO 39

C      WHENEVER THE NEVC ENTRY OF A ROW IS REACHED, COMPUTE
C      ITS NEW VALUE THUS.
C
38 TTE(K,NC)=FIX(-T2(NO)/PIV)
39 CONTINUE
   GO TO 42
40 DO 41 NC=1,N

C      ONCE THE NDVR ELEMENT IS FOUND AGAIN, COMPUTE ITS
C      NEW VALUE THUS.
C
   TTE(K,NC)=FIX(T1(NC+M)/PIV)

C      FOR THE NEVC ELEMENT OF THE NDVR ROW, DIVIDE 1.0
C      BY THE CURRENT PIVOT ELEMENT.
C
   IF(NC.EQ.NE.AND.NN.EQ.NT) TTE(K,NC)=FIX(1.0/PIV)
41 CONTINUE

```

```

42 CONTINUE
  WRITE (NSCRAT) ((TTE(L1,L2),L2=1,N),L1=1,K3)
43 CONTINUE
44 CONTINUE

      LET NTAPE NOW REFER TO THE TAPE WITH THE MOST CURRENT
      TTE VALUES.

      NTAPE=NSCRAT
      TR(NDVR)=FIX(TB(NDVR)/PIV)

      RECOMPUTE INDEX ROWS 1 THRU NROW

      IF(UNIT(3)) 45,55,55
45 CALL CINDX(NROW)
55 RETURN
      END

```

```

C
C
C
C

C
C
C

```



```

C C C C
FUNCTION FIX(7)
C
C      THIS FUNCTION CHANGES FLOATING POINT VALUES WITHIN
C      10.**(-L) OF AN INTEGER TO THAT INTEGER.
C
C      COMMON /PAR/ L
C      Y=50./10.**L
C      M=L-1
C      X=.1
C
C      IN THE DO 1 LOOP, X WILL VARY FROM 1.0 TO
C      10.**(L-2)
C
C      DO 1 N=1,M
C      X=10.**X
C      I=F=X+7
C      J=I-2
C      DO 1 K=1,3
C      G=J+K
C
C      IF THE ABSOLUTE VALUE OF F-G IS GREATER THAN
C      50./10.**L, RETURN THE VALUE OF Z.
C
C      IF (ABS(F-G)-Y) 2,2,1
C      1 CONTINUE
C      FIX=Z
C      RETURN
C      2 FIX=G/X
C      RETURN
C      END
C C C C

```

```

C C
C C
C C
C C
C C
C C
SUBROUTINE ALTST(NPPT,NROW)

SUBROUTINE ALTST (ALTERNATE SOLUTION TEST) CHECKS
THE SOLVED TABLEAU FOR ALTERNATE VALID SOLUTIONS.
IF ONE OR MORE VALID SOLUTIONS EXISTS, ALTST
GENERATES AND OUTPUTS THEM.

COMMON TTE(20,390),TT(5,724),TB(344),TL(344,6),JCOL(724,2),
1 JROW(344,2),JN(6,3),JA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2 NTT,NTAPE,INDEX,NROW1

C C
C C
C C
C C
C C
T2 IS THE ARRAY OF NC VALUES TO BE USED IN PERM.
TEMP STORES THE T2 VALUES TEMPORARILY WHILE PERM
IS CALLED THE FIRST TIME.

COMMON /DATA1/ T2(344)
DIMENSION TEMP(344)
DO 14 NC=1,NCOL

C C
C C
C C
C C
THE INDEX ROWS ARE TESTED FOR AN ALL-ZERO COLUMN,
INDICATING ALTERNATE SOLUTIONS ON THAT COLUMN.

DO 1 NP=1,NPRI
IF(TT(NP,NC).NE.0) GO TO 14
1 CONTINUE

C C
C C
C C
PUT THE VALUES OF THE NC COLUMN IN T2.

N=NVAR
NO=0
NC1=NC
IF(NC.LE.NVAR) GO TO 2

C C
C C
C C
IF NC1 EXCEEDS NVAR, LET NC1=NC-NVAR AND SKIP
THE FIRST HALF OF THE FILE.

```

```

NC1=NC1-NVAR
NT=NBLK/2
N=NOBJ
2 REWIND NTAPE
  IF(NC.LE.NVAR) GO TO 4
  DO 3 I1=1,NT
3 READ (NTAPE)

C      READ THE FILE SEQUENTIALLY TO GET THE T2 VALUES.
C
C
4 DO 5 NN=1,8
  K2=IN(NN,2)
  K3=INDEX
  J=IN(NN,3)
  DO 5 LL=1,J
  IF(LL.EQ.J.AND.K2.NE.0) K3=K2
  READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
  DO 5 K=1,K3
  NO=NO+1
5 T2(NO)=TTE(K,NC1)
  DO 13 NO=1,NOBJ
  IF(TB(NO).LE.0..OR.T2(NO).LE.0) GO TO 13

C      IF A POSITIVE VALUE OF TE IS FOUND AND THE TB
C      VALUE CORRESPONDING TO IT IS POSITIVE, PERM
C      IS CALLED TO PIVOT ON THAT TE ELEMENT.
C
  CALL PERM(NC,NO,NROW)
  DO 6 NB=1,NOBJ
  IF(TB(NB).LT.0.) GO TO 7

C      IF THE NEW TABLEAU HAS NO NEGATIVE TB ELEMENTS,
C      THE ALTERNATE SOLUTION IS VALID.
C
6 CONTINUE
  CALL POUT(NPRT)

```

```

C
C
C
C
C
C
C
      PERM IS CALLED AGAIN TO RETURN THE TABLEAU TO ITS
      ORIGINAL FORM FOR FURTHER ALTERNATE SOLUTION SEARCH.
      A SECOND T2 ARRAY HAS TO BE CREATED BEFORE PERM IS
      ENTERED AGAIN.
      7 DO 8 NNO=1,NOBJ
      8 TEMP(NNO)=T2(NNO)
      M=0
      REWIND NTAPE
      IF(NC.LE.NVAR) GO TO 10
      DO 9 I1=1,NT
      9 READ (NTAPE)
      10 DO 11 NN=1,8
      K2=IN(NN,2)
      K3=INDEX
      J=IN(NN,3)
      DO 11 LL=1,J
      IF(LL.EQ.J.AND.K2.NE.0) K3=K2
      READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
      DO 11 K=1,K3
      M=M+1
      11 T2(M)=TTE(K,NC1)
      CALL PERM(NC,NO,NROW)
      DO 12 NNO=1,NOBJ
      12 T2(NNO)=TEMP(NNO)
      13 CONTINUE
      14 CONTINUE
      RETURN
      END

```

```

C C SUBROUTINE POUT(NPRT)
C C SUBROUTINE POUT (PRINT OUT) FORMS AND PRINTS THE
C C SOLUTION INFORMATION WITH CODED VARIABLE NAMES AND
C C WITH MESSAGES INDICATING ALTERNATE SOLUTIONS.
C C
COMMON TTE(20,390),TT(6,724),TB(344),TL(344,6),JCOL(724,2),
1 JROW(344,2),IN(8,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2 NTT,NTAPE,INDEX,NROW1
COMMON /PAR/ L

C C X1 IS THE ARRAY OF LABELS FOR THE DECISION
C C VARIABLES. WORK IS THE TEMPORARY ARRAY FOR THE
C C PRINTOUT OF THE FINAL SOLUTION. THE FORMAT OF
C C THE FINAL SOLUTION IS SUBSCRIPT NUMBER, LABEL,
C C VALUE OF THE DECISION VARIABLE, VALUES OF THE
C C POSITIVE AND NEGATIVE DEVIATIONS, RESPECTIVELY, AND
C C THE VALUES OF THE ACHIEVEMENT FUNCTION.
C C
COMMON /LABELS/ X1(380,2)
COMMON WORK(380,4)
NPRT=NPRT+1
IF(NPRT.NE.1) WRITE(6,31) NPRT

C C THE OUTPUT ARRAY IS ZEROED.
C C
DO 12 I=1,NVAR
DO 12 J=1,4
12 WORK(I,J)=0.
C THE OUTPUT ARRAY IS FILLED.
DO 13 NP=1,NPRI
13 WORK(NP,1)=FIX(TA(NP))

C C THE FUNCTION FIX RETURNS VALUES THAT HAVE STRAYED
C C FROM BEING INTEGER VALUED DUE TO THE PRECISION
C C LIMITATIONS OF THE COMPUTER, BACK TO INTEGERS.

```


C

```
DO 14 NO=1,NORJ
IC=JROW(NO,1)
IF=JROW(NO,2)
```

C

```
14 WORK(IR,IC)=FIX(TB(NO))
THE OUTPUT ARRAY IS WRITTEN OUT.
```

```
WRITE(6,33) NVAR,NORJ,NORJ,NPRI
```

```
I=MAXC(NPRI,NVAR,NORJ)
```

```
DO 20 K=1,I
```

```
IF(K.GT.NPRI)GO TO 16
```

```
WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4),WORK(K,1)
```

```
GO TO 20
```

```
16 IF(K.GT.NORJ)GO TO 17
```

```
WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4)
```

```
GO TO 20
```

```
17 WRITE(6,34) K,(X1(K,M),M=1,2),WORK(K,2)
```

```
20 CONTINUE
```

```
31 FORMAT(/," ALTERNATE SOLUTION NUMBER ",I3)
```

```
33 FORMAT(/," SUBSCRIPT",5X,"VARIABLE CODE",5X,I6," TERMS XSTAR",  
1 1X,I6," TERMS PSTAP",1X,I6," TERMS NSTAR",2X,I6," TERMS ZBAR")
```

```
34 FORMAT(" ",I6,5X,2A10,4F19.8)  
RETURN
```

```
ENTRY ERRORS
```

```
WRITE(6,35) NPRT
```

```
35 FORMAT(/," **ALGORITHM DID NOT FINISH AFTER ",I3," ITERATIONS**")  
RETURN  
END
```

```

C
C
C
C
SUBROUTINE STORE(NROW)

C
C
C
C
THIS SUBROUTINE STORES THE VALUES OF SYSTEM PARAMETERS
FOR FUTURE USE.

COMMON TTE(20,380), TT(6,724), T3(344), T1(344,6), JCOL(724,2),
1     JROW(344,2), JN(8,3), TA(5), NOBJ, NPRI, NVAR, NCOL, NBLK,
2     NTT, NTAPE, INDEX, NROW1
COMMON /LABELS/ X1(380,2)
NROW1=NROW
REWIND 3
BUFFER IN (3,1) (TT(1,1), TT(NPRI, NCOL))

C
C
C
C
GENERATE THE MOST CURRENT TTE MATRIX ON TAPE8
PRIOR TO STORING OTHER INFO AND WAITING.

REWIND NTAPE
REWIND 8
DO 5 NT=1,2
N=NORJ
IF(NT.EQ.1) N=NVAR
DO 5 I=1,8
K2=IN(I,2)
K3=INDEX
J=IN(I,3)
DO 5 LL=1,J
IF(LL.EQ.J.AND.K2.NE.0) K3=K2
5  READ (NTAPE) ((TTE(L1,L2), L2=1,N), L1=1,K3)
15 WRITE (8) ((TTE(L1,L2), L2=1,N), L1=1,K3)
IF(UNIT(3)) 15,50,50
15 REWIND 4

C
C
C
C
STORE ON TAPE4 THE REMAINING INFORMATION NECESSARY TO
RESUME OPERATIONS LATER.

NV=NVAR

```

```

      BUFFER OUT (4,1) (TT(1,1),NROW1)
      IF(UNIT(4)) 20,50,50
20  BUFFER OUT (4,1) (X1(1,1),X1(VV,2))
30  IF(UNIT(4)) 30,50,50
      ENDFILE 4
      STOP
50  WRITE(6,60)
60  FORMAT("ERROR DETECTED IN SUBROUTINE STORE")
      STOP "ERROR WITH STORE"
      END

```

```

PROGRAM MAIN3(INPUT=3008/80,OUTPUT=30008,TAPE1=240008,TAPE8=167008
1,TAPE2=240009,TAPE3=08,TAPE4=03,TAPE5=INPUT,TAPE6=OUTPUT)

```

```

C THE COMPUTATIONAL TABLEAU IS BROKEN INTO FIVE ARRAYS:

```

```

C TTE(NO,NC) = EQUATION FIELD
C TR(NO) = RIGHT HAND SIDE VALUES
C TT(NP,NC) = TOP SUB AND INDEX ROWS
C TL(NO,NP) = LEFT SUB VALUES
C TA(NP) = GOAL ACHIEVEMENT VECTOR

```

```

C TT IS A STORAGE ARRAY FOR TOP SUB AND INDEX VALUES; ONE
C OR THE OTHER (AND NOT BOTH) IS JEDED IN MEMORY AT ONE
C TIME. ONLY IN SUBROUTINE CINDX DO BOTH SETS INTERACT; HOWEVER,
C INDEX ROWS ARE COMPUTED DIRECTLY FROM TOP SUB VALUES. AFTER
C THIS COMPUTATION, THE TOP SUB VALUES ARE NOT NEEDED AGAIN
C UNTIL CALLED FOR BY ROUTINE PERM OR CINDX.

```

```

C THE SUBSCRIPTS ABOVE INDICATE:

```

```

C NC=1,NOBJ THE NUMBER OF OBJECTIVE EQUATIONS
C NC=1,NCOL THE NUMBER OF DECISION VARIABLES PLUS THE
C           NUMBER OF OBJECTIVE EQUATIONS
C           (FOR TTE MATRIX, NC=1, MAX(NOBJ,NVAR) )
C NP=1,NPRI THE NUMBER OF PRIORITIES

```

```

C THE COLUMN AND ROW HEADINGS FOR THE FINAL PRINTOUT ARE
C ENCODED INTO THE JCOL(NSUB,NTYPE) AND JROW(NSUB,NTYPE)
C FIELDS.

```

```

C NSUB = 2 IMPLIES A DECISION VARIABLE X
C       = 3 IMPLIES A POS. DEVIATION P
C       = 4 IMPLIES A NEG. DEVIATION N

```

```

C ARRAY IN(8,3) MAINTAINS THE INFORMATION ON HOW TAPE1 IS
C INITIALLY CREATED. FOR EACH CONSTRAINT SET I,

```

```

C IN(I,1) IS THE NUMBER OF EQUATIONS IN THE SET.

```

```

C IN(I,2) IS THE REMAINDER OF IN(I,1)/INDEX

```

```

C IN(I,3) IS THE NUMBER OF BLOCKS CREATED FOR THE SET.

```

```

C C C SCALARS IN THE COMMON AREA ARE
C C C NOBJ = NUMBER OF OBJECTIVE EQUATIONS
C C C NPRI = NUMBER OF PRIORITY LEVELS
C C C NVAR = NUMBER OF DECISION VARIABLES
C C C NCOL = NOBJ+NVAR
C C C NBLK = TOTAL NUMBER OF BLOCKS ON TAPE1
C C C NTT = SWITCH FOR INDICATING TOP STUB OR INDEX
C C C ROWS (1 FOR TOP STUB AND 2 FOR INDEX ROWS)
C C C NTAPE = SWITCH FOR INDICATING FILE OF CURRENT TTE
C C C VALUES (1 FOR TAPE1 AND 2 FOR TAPE2)
C C C INDEX = MAXIMUM NUMBER OF POWS PER BLOCK ON TAPE1.
C C C NROW1 = THE PRIORITY LEVEL THE ALGORITHM IS USING
C C C PRIOR TO ANOTHER CALL TO SUBROUTINE STORE.
C C C
C C C COMMON TTE(20,380),TT(6,724),TB(344),TL(344,6),JCOL(724,2),
C C C 1 JROW(344,2),IN(8,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
C C C 2 NTT,NTAPE,INDEX,NROW1
C C C
C C C L IS A POSITIVE INTEGER GREATER THAN 1 WHICH
C C C SPECIFIES THE NUMBER OF DECIMAL PLACES OF PRE-
C C C CISION USED IN FUNCTION FIX.
C C C
C C C IMAX IS THE MAXIMUM NUMBER OF TAPEAU GENERATIONS
C C C ALLOWED BEFORE A RECURSIVE LOOP IS ASSUMED.
C C C NPRT IS THE NUMBER OF PRINTOUTS PROCESSED.
C C C
C C C COMMON /PAR/ L
C C C DATA IMAX,NPRI/400,0/
C C C T1=SECOND(T)
C C C
C C C SUBROUTINE RESTOR RESTORES THE PARAMETERS NECESSARY
C C C TO CONTINUE THE ALGORITHM. LROW IS THE PRIORITY LEVEL
C C C THAT THE PROGRAM SHOULD BEGIN WITH ON THIS RUN.
C C C
C C C CALL RESTOR(LROW)

```



```

C C C C C C      WHEN NROW BECOMES NPRI+1, THE ALGORITHM FOR DETER-
C C C C C C      MINING A FINAL SOLUTION IS ENDED.  NROW IS THE
C C C C C C      ROW THAT THE PROGRAM IS CURRENTLY USING IN THE
C C C C C C      MULTIPHASE OPTIMIZING ROUTINE.

C C C C C C      DO 4 NROW=LROW,NPRI
C C C C C C      PRINT *, "THE PRIORITY LEVEL IS ", NROW

C C C C C C      CALLING CINDX(NROW) CAUSES THE INDEX ROWS 1,2,...,NROW
C C C C C C      TO BE COMPUTED.

C C C C C C      CALL CINDX(NROW)

C C C C C C      SUBROUTINE TEST COMPUTES THE ENTERING VARIABLE
C C C C C C      COLUMN (NEVC) AND THE DEPARTING VARIABLE ROW (NDVR).
C C C C C C      WHEN NEVC=0, THE ROW CANNOT BE OPTIMIZED FURTHER.

C C C C C C      CALL TEST(NEVC,NDVR,NROW), RETURNS(5)
C C C C C C      DO 3 IVAL=1,IMAX
C C C C C C      PRINT *, "IVAL,NDVR, AND NEVC ARE ", IVAL, " ", NDVR, " ", NEVC
C C C C C C      IF(NEVC.LE.0) GO TO 4

C C C C C C      SUBROUTINE PERM COMPUTES THE NEW TABLEAU.

C C C C C C      CALL PERM(NEVC,NDVR,NROW)
C C C C C C      IF(SECOND(T)-T1.LE.3000.0) GO TO 3
C C C C C C      CALL POUT(NPRI)
C C C C C C      CALL STORE(NROW)
C C C C C C      3 CALL TEST(NEVC,NDVR,NROW), RETURNS(5)

C C C C C C      THE DO 3 LOOP ENDS BY REACHING ITS UPPER LIMIT
C C C C C C      WITHOUT CONVERGING.

C C C C C C      CALL ERRORS(IMAX)
C C C C C C      GO TO 5

```

```

4 CONTINUE
  NROW=NPRI
  SUBROUTINE POUT IS CALLED TO PRINT OUT THE DESIRED
  SOLUTION.
  CALL POUT(NPRT)
  SUBROUTINE ALTST TESTS FOR ALTERNATIVE VALID SOLUTIONS
  AND PRINTS OUT ANY THAT IT PRODUCES.
  CALL ALTST(NPRT,NROW)
  STOP
5 STOP "FAILURE"
  END

```

```

C
C
C
C
C
C
C
C

```

SUBROUTINE CINDX(NROW)

THE FUNCTION OF CINDX IS TO COMPUTE INDEX ROWS.
CINDX DOES THIS FROM ROW 1 TO ROW NROW.
THE TOP STUB VALUES MUST BE IN MEMORY. HENCE,
CHECK TO SEE IF NTT=1. IF NOT, THE PROGRAM READS
IN RECORD 1 OF TAPE3 AND SETS NTT=1.

```
COMMON TTE(20,380),TT(6,724),TB(344),TL(344,6),JCOL(724,2),
1  JROW(344,2),IN(6,3),TA(5),NOBJ,NPRI,NVAR,NCOL,NBLK,
2  NTT,NTAPE,INDEX,NROW1
  IF(NTT.EQ.1) GO TO 2
  REWIND 3
  BUFFER IN (3,1) (TT(1,1),TT(NPRI,NCOL))
  IF(UNIT(3)) 1,11,11
1  NTT=1
2  DO 9 NP=1,NROW
  TA(NP)=0.
```

THE RIGHT HAND SIDE OF THE INDEX ROWS ARE COMPUTED.

```
DO 3 NO=1,NOBJ
3  TA(NP)=TA(NP)+TB(NO)*TL(NO,NP)
DO 4 NC=1,NCOL
4  TT(NP,NC)=-TT(NP,NC).
  REWIND NTAPE
```

CINDX NEEDS EVERY ELEMENT OF THE TTE MATRIX FOR
COMPUTING INDEX ROWS. THE TAPE (NTAPE) IS READ
SEQUENTIALLY--FIRST, THE ORIGINAL LEFT HAND
SIDE AND SECOND, THE ORIGINAL RIGHT HAND SIDE.

NO IS A COUNTER RANGING FROM 1 TO NOBJ. IT IS
INITIALLY SET TO 0 AND SPECIFIES THE NUMBER OF
COLUMNS TO BE READ IN. FOR LEFT HAND TTE VALUES,
N=NVAR; FOR RIGHT HAND VALUES, N=NOBJ.

```

C C C C C C C
C SIMILARLY M IS AN INCREMENT FOR THE SECOND
C COEFFICIENT OF TT. FOR THE LEFT HAND TTE VALUES,
C M=0, THE COLUMNS OF TT AND TTE MATCH. FOR THE
C RIGHT HAND TTE VALUES, M=NVAR. IN THE LATTER
C CASE TTE COLUMNS RANGE FROM 1 TO NCBJ AND TT COLUMNS
C RANGE FROM NVAR+1 TO NVAR+NCBJ(=NCOL).
C
DO 8 NT=1,2
N=NVAR
M=NO=0
IF (NT.EQ.1) GO TO 5
M=NVAR
N=NCBJ
5 DO 7 I=1,8
K2=IN(I,2)
K3=INDEX
C K3 REFERS TO THE NUMBER OF ROWS IN A CERTAIN BLOCK OF TTE.
C J=IN(I,3)
C J REFERS TO THE NUMBER OF BLOCKS IN CONSTRAINT SET I.
DO 7 L=1,J
IF (L.EQ.J.AND.K2.NE.0) K3=K2
READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
DO 6 K=1,K3
NC=NO+1
DO 6 NC=1,N
6 TT(NP,NC+M)=TT(NP,NC+M)+TTE(K,NC)*TL(NO,NP)
7 CONTINUE
8 CONTINUE
9 CONTINUE
NTT=2
11 RETURN
END

```


C C FILE AND READ FROM THE MIDDLE OF THE FILE ONWARD.

```

NT=NBLK/2
N=NOBJ
DO 4 NT1=1,NT
4 READ (NTAPE)
5 DO 11 I=1,8
K2=IN(I,2)
K3=INDEX
J=IN(I,3)
DO 10 LL=1,J

```

C C ONLY FOR THE FINAL BLOCK IF A CONSTRAINT SET IS THE
C C NUMBER OF ROWS EQUAL TO K2=IN(I,2).

```

IF (LL.EQ.J.AND.K2.NE.0) K3=K2
READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
DO 9 K=1,K3
NC=NO+1
T2(NO)=TTE(K,NE)
IF (T2(NO).LE.0) GO TO 9
V=TB(NO)/T2(NC)
IF (NDVR.EQ.0) GO TO 8
IF (V-VDVR) 8,5,9
6 DO 7 NP=1,NPRI
7 CONTINUE
8 VDVR=V
NDVR=NO
9 CONTINUE
10 CONTINUE
11 CONTINUE
IF (NDVR.GE.1) RETURN
WRITE(6,20)
20 FORMAT (" *PROGRAM TERMINATED--FAILED-PIVOT COMPUTATION**")
RETURN ERR1

```


C C C C C
 IS LOCATED IN THE BLOCK NJ1BER IB+NBLK/2. LL IS
 THE NUMBER OF ROWS OF CONSTRAINT SET NT MINUS NDVR.
 WHEN LL IS NO LONGER NEGATIVE, THE CONSTRAINT SET NT
 HOLDS THE NDVR ROW.

```

DO 30 NT=1,8
K=IN(NT,2)
LL=IN(NT,1)-N
IF(LL.GE.0) GO TO 31
N=-LL
IB=IB+IN(NT,3)
30 CONTINUE
31 IB=IB+N/INDEX
N1=MOD(N,INDEX)
IF(N1.NE.0) IB=IB+1
IF(N1.EQ.0) N1=INDEX

```

C C C C C C
 J1 IS THE NUMBER OF ROWS IN BLOCK IB.
 N1 IS THE NDVR POW IN BLOCK IB. WITH THIS CORE-
 SWAPPING ROUTINE, N1 IS THE CORRECT ROW COEFFICIENT
 OF THE TTE MATRIX.

```

J1=INDEX
IF(LL.LT.K) J1=K
II=IB-1
REWIND NTAPE
DO 36 NT=1,2
M=0
N=NVAR
IF(NT.NE.2) GO TO 32
M=NVAR
N=NOBJ
II=NBLK/2-1

```

C C C
 CHECK TO SEE IF IB=1 (OR EQUIVALENTLY II=0).
 WE WANT TO SKIP II RECORDS ON THE TAPE. FOR II=0,

```

C      WE MERELY READ THE INITIAL RECORD.
C
32 IF (II.EQ.0) GO TO 34
   DO 33 IJ=1,II
33 READ (NTAPE)
34 READ (NTAPE) ((TTE(I1,I2),I2=1,N),I1=1,J1)
   DO 35 NC=1,N
35 T1(NC+M)=TTE(N1,NC)
36 CONTINUE

C      THE PIVOT ELEMENT IS TTE(NDVR,NEVC)=T1(NEVC).
C
   PIV=T1(NEVC)
   PIB=TB(NDVR)
   NN=1

C      NN REFERS TO THE SIDE OF THE TTE MATRIX WHERE NEVC
C      IS LOCATED. WHEN NEVC EXCEEDS NVAR, NN=2.
C      OTHERWISE, NN=1.
C
   NE=NEVC
   IF(NEVC.GT.NVAR) NE=NE-NVAR
   IF(NEVC.GT.NVAR) NN=2
   REWIND NTAPE
   NSCRAT=NTAPE+1

C      NSCRAT REFERS TO THE SCRATCH TAPE ON WHICH THE LATEST
C      TTE VALUES ARE WRITTEN.
C
   IF(NSCRAT.GT.2) NSCRAT=1
   REWIND NSCRAT
   DO 44 NT=1,2
   N=NVAR
   M=NO=0
   IF(NT.EQ.1) GO TO 37
   M=NVAR

```



```

42 CONTINUE
   WRITE (NSCRAT) ((TTE(L1,L2),L2=1,N),L1=1,K3)
43 CONTINUE
44 CONTINUE

      LET NTAPE NOW REFER TO THE TAPE WITH THE MOST CURRENT
      TTE VALUES.

      NTAPE=NSCRAT
      TB(NDVR)=FIX(TB(NDVP)/PIV)

      RECOMPUTE INDEX ROWS 1 THRU NROW

      IF(UNIT(3)) 45,55,55
45 CALL CINDX(NROW)
55 RETURN
      END

```

```

C
C
C
C
C
C
C
C

```

```

C
C
C
C
      FUNCTION FIX(Z)
      THIS FUNCTION CHANGES FLOATING POINT VALUES WITHIN
      10.**(-L) OF AN INTEGER TO THAT INTEGER.

      COMMON /PAR/ L
      Y=50./10.**L
      M=L-1
      X=.1

C
C
C
C
      IN THE DO 1 LOOP, X WILL VARY FROM 1.0 TO
      10.**(L-2)

      DO 1 N=1,M
      X=10.**X
      I=F=X*Z
      J=I-2
      DO 1 K=1,3
      G=J+K

C
C
C
C
      IF THE ABSOLUTE VALUE OF F-G IS GREATER THAN
      50./10.**L, RETURN THE VALUE OF Z.

      IF(ABS(F-G)-Y) 2,2,1
      1 CONTINUE
      FIX=Z
      RETURN
      2 FIX=G/X
      RETURN
      END

```

```

C
C
C
C
C
C
SUBROUTINE ALTST(NPRT,NROW)

C
C
C
C
C
C
SUBROUTINE ALTST (ALTERNATE SOLUTION TEST) CHECKS
THE SOLVED TABLEAU FOR ALTERNATE VALID SOLUTIONS.
IF ONE OR MORE VALID SOLUTIONS EXISTS, ALTST
GENERATES AND OUTPUTS THEM.

C
C
C
COMMON TIE(20,330), IT(6,724), IB(344), IL(344,6), JCOL(724,2),
1      JROW(344,2), IN(8,3), TA(5), NOBJ, NPRI, NVAR, NCOL, NBLK,
2      NTT, NTAPE, INDEX, NROW1

C
C
C
C
C
C
T2 IS THE ARRAY OF NC VALUES TO BE USED IN PERM.
TEMP STORES THE T2 VALUES TEMPORARILY WHILE PERM
IS CALLED THE FIRST TIME.

C
C
C
C
COMMON /DATA1/ T2(344)
DIMENSION TEMP(344)
DO 14 NC=1,NCOL

C
C
C
C
C
C
THE INDEX ROWS ARE TESTED FOR AN ALL-ZERO COLUMN,
INDICATING ALTERNATE SOLUTIONS ON THAT COLUMN.

C
C
C
C
DO 1 NP=1,NPRI
IF(TT(NP,NC).NE.0) GO TO 14
1 CONTINUE

C
C
C
C
PUT THE VALUES OF THE NC COLUMN IN T2.

C
C
C
N=NVAR
NO=6
NC1=NC
IF(NC.LE.NVAR) GO TO 2

C
C
C
C
IF NC1 EXCEEDS NVAR, LET NC1=NC-NVAR AND SKIP
THE FIRST HALF OF THE FILE.

```

```

NC1=NC1-NVAR
NT=NBLK/2
N=NORJ
2 REWIND NTAPE
  IF(NC.LE.NVAR) GO TO 4
  DO 3 I1=1,NT
3 READ (NTAPE)

C
C
C      READ THE FILE SEQUENTIALLY TO GET THE T2 VALUES.

4 DO 5 NN=1,8
  K2=IN(NN,2)
  K3=INDEX
  J=IN(NN,3)
  DO 5 LL=1,J
  IF(LL.EQ.J.AND.K2.NE.0) K3=K2
  READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
  DO 5 K=1,K3
  NC=NO+1
5 T2(NO)=TTE(K,NC1)
  DO 13 NO=1,NORJ
  IF(TB(NO).LE.0..OR.T2(NO).LE.0) GO TO 13

C
C
C      IF A POSITIVE VALUE OF TE IS FOUND AND THE TB
C      VALUE CORRESPONDING TO IT IS POSITIVE, PERM
C      IS CALLED TO PIVOT ON THAT TE ELEMENT.
C

  CALL PERM(NC,NO,NROW)
  DO 6 NB=1,NORJ
  IF(TB(NB).LT.0.) GO TO 7

C
C
C      IF THE NEW TABLEAU HAS NO NEGATIVE TB ELEMENTS,
C      THE ALTERNATE SOLUTION IS VALID.
C

6 CONTINUE
  CALL POUT(NPRT)

```



```

C
C
C
C
C
C
C
      PERM IS CALLED AGAIN TO RETURN THE TABLEAU TO ITS
      ORIGINAL FORM FOR FURTHER ALTERNATE SOLUTION SEARCH.
      A SECOND T2 ARRAY HAS TO BE CREATED BEFORE PERM IS
      ENTERED AGAIN.

      7 DO 8 NNO=1,N0BJ
      8 TEMP(NNO)=T2(NNO)
      M=0
      REWIND NTAPE
      IF(NC.LE.NVAR) GO TO 10
      DO 9 I1=1,NT
      9 READ (NTAPE)
      10 DO 11 NN=1,8
      K2=IN(NN,2)
      K3=INDEX
      J=IN(NN,3)
      DO 11 LL=1,J
      IF(LL.EQ.J.AND.K2.NE.0) K3=K2
      READ (NTAPE) ((TTE(L1,L2),L2=1,N),L1=1,K3)
      DO 11 K=1,K3
      M=M+1
      11 T2(M)=TTE(K,N01)
      CALL PERM(NC,N0,NROW)
      DO 12 NNO=1,N0BJ
      12 T2(NNO)=TEMP(NNO)
      13 CONTINUE
      14 CONTINUE
      RETURN
      END

```



```

C
DO 14 NO=1, NOBJ
IC=JROW(NO,1)
IF=JROW(NO,2)
14 WORK(IR,IC)=FIX(I3(NO))
    THE OUTPUT ARRAY IS WRITTEN OUT.
    WRITE(6,33) NVAR,NOBJ,NPRI
    I=MAX0(NPRI,NVAR,NOBJ)
    DO 20 K=1,I
    IF(K.GT.NPRI)GO TO 16
    WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4),WORK(K,1)
    GO TO 20
16 IF(K.GT.NOBJ)GO TO 17
    WRITE(6,34) K,(X1(K,M),M=1,2),(WORK(K,J),J=2,4)
    GO TO 20
17 WRITE(6,34) K,(X1(K,M),M=1,2),WORK(K,2)
20 CONTINUE
31 FORMAT(/," ALTERNATE SOLUTION NUMBER ",I3)
33 FORMAT(/," SJSCRIPT",5X,"VARIABLE CODE",5X,I6," TERMS XSTAR",
1 1X,I6," TERMS PSTAR",1X,I6," TERMS NSTAR",2X,I6," TERMS ZBAR")
34 FORMAT(" ",I6,5X,2A10,4F19.8)
RETURN
ENTRY ERRORS
WRITE(6,35) NPRT
35 FORMAT(/," **ALGORITHM DID NOT FINISH AFTER ",I3," ITERATIONS**")
RETURN
END

```

```

C
C
C
C
SUBROUTINE STORE(NROW)

THIS SUBROUTINE STORES THE VALUES OF SYSTEM PARAMETERS
FOR FUTURE USE.

COMMON TTE(20,380), TT(6,724), T3(344), TL(344,6), JCOL(724,2),
1 JROW(344,2), IN(8,3), TA(5), NOBJ, NPRI, NVAR, NCOL, NBLK,
2 NTT, NTAPE, INDEX, NROW1
COMMON /PAR/ L
COMMON /LABELS/ X1(380,2)
NROW1=NROW
REWIND 3
BUFFER IN (3,1) (TT(1,1), TT(NPRI, NCOL))

GENERATE THE MOST CURRENT TTE MATRIX ON TAPE8
PRIOR TO STORING OTHER INFO AND HALTING.

REWIND NTAPE
REWIND 8
DO 5 NT=1,2
N=NOBJ
IF (NT.EQ.1) N=NVAR
DO 5 I=1,8
K2=IN(I,2)
K3=INDEX
J=IN(I,3)
DO 5 LL=1,J
IF (LL.EQ.J.AND.K2.NE.0) K3=K2
READ (NTAPE) ((TTE(L1,L2), L2=1,N), L1=1,K3)
5 WRITE (8) ((TTE(L1,L2), L2=1,N), L1=1,K3)
IF (UNIT(3)) 15,50,50
15 REWIND 4

STORE ON TAPE4 THE REMAINING INFORMATION NECESSARY TO
RESUME OPERATIONS LATER.
C
C
C
C

```

```

NV=NVAR
  BUFFER OUT (4,1) (TT(1,1),NROW1)
  IF (UNIT(4)) 20,50,50
20  BUFFER OUT (4,1) (X1(1,1),X1(VV,2))
30  IF (UNIT(4)) 30,50,50
  ENDFILE 4
  STOP
50  WRITE(6,60)
60  FORMAT("ERROR DETECTED IN SUBROUTINE STORE")
  STOP "ERROR WITH STORE"
  ENTRY RESTOR

C      THIS ENTRY POINT RESTOR(ES) THE VALUES OF SYSTEM
C      PARAMETERS TO BEGIN THE OPERATIONS AGAIN.
C
  READ(5,*) L
  REWIND 3
  BUFFER IN (4,1) (TT(1,1),NROW1)
  IF (UNIT(4)) 110,140,140
110  BUFFER IN (4,1) (X1(1,1),X1(NVAR,2)) /
  BUFFER OUT (3,1) (TT(1,1),TT(VPRI,NCOL))
  IF (UNIT(4)) 120,140,140
120  NROW=NROW1
  NNT=NTAPE=1
  REWIND NTAPE
  DO 135 NT=1,2
  N=NOBJ
  IF (NT.EQ.1) N=NVAR
  DO 135 I=1,8
  K2=IN(I,2)
  K3=INDEX
  J=IN(I,3)
  DO 135 LL=1,J
  IF (LL.EQ.J.AND.K2.NE.0) K3=K2
  READ (8) (TTE(L1,L2),L2=1,N),-1=1,K3)
135  WRITE (NTAPE) (TTE(L1,L2),L2=1,N),L1=1,K3)

```


IF(UNIT(3)) 139,140,140
138 RETURN
140 STOP "ERROR IN RESTORE"
END

APPENDIX C
THE PATTERN SEARCH ALGORITHM

C	SIGN	INDICATES POSITIVE OR NEGATIVE
C		DEVIATION TO BE MINIMIZED FROM
C	IRON	GOAL EQUATION IRON AT MPRI
C		LEVEL WITH WEIGHT FACTOR WEIGHT
C		GOAL EQUATION OF GIVEN
C		DEVIATION VARIABLE
C	Y	VECTOR OF FUNCTIONS OF DECISION
C		VARIABLES
C	XBASE	BASE POINT FOR HOCKE-JEEVES
C		ALGORITHM
C	ZTEST	ACHIEVEMENT (OBJECTIVE) FUNCTION VALUE VECTOR
C	C	NUMBER OF SKILL TYPES (INTEGER)
C	T	NUMBER OF TIME PERIODS (INTEGER)
C	GAMMA	OVERSEAS THEATER SURVIVAL RATE
C	IDEL	LENGTH OF TECHNICAL TRAINING
C	SIR	DESIRED TRAINEE PER INSTRUCTOR RATIO
C	PCT	MAXIMUM FRACTION OF INSTRUCTORS
		PREMITTED TO RETURN TO
		THE CONUS FORCE
C	ITAU	LENGTH OF BASIC TRAINING
C	XBETA	SURVIVAL RATE IN BASIC TRAINING
C	SVLTT	SURVIVAL RATE IN TECHNICAL TRAINING
C	CPT	MINIMUM FRACTION OF CONUS GOAL
C		ACHIEVEMENT AUTHORIZED
C	TFG	OVERSEAS THEATER GOAL
C	TFI	INITIAL OVERSEAS THEATER LEVEL
C	CFG	CONUS FORCE GOAL
C	CFI	INITIAL CONJS FORCE LEVEL
C	W	FLOW RATE OF PEOPLE INTO TECHNICAL
C		TRAINING
C	TII	INITIAL TECHNICAL TRAINING INSTRUCTOR
C		LEVEL

ALL REQUIRED DATA IS INPUT USING LIST DIRECTED
(UNFORMATED) READ STATEMENTS.

```

C THE SHIFT FUNCTION IS USED FOR MULTIPLICATION AND
C DIVISION BY POWERS OF 2.
C FOR EXAMPLE, SHIFT(X,N)=X*(2**N) WHERE N IS ANY
C INTEGER IN THE INTERVAL (-50,50).
C
C SUBROUTINE YVALUE IS SET UP TO EVALUATE THE
C FUNCTIONS OF THE DECISION VARIABLES -- ONE
C FOR EACH GOAL EQUATION.
C
C      DIMENSION X(300),Z(7)
C      COMMON/COMM01/NVAR,NOBJ,NPRIOR,NACH,NMAX
C      COMMON/COMM02/X
C      COMMON/COMM08/Z
C      CALL DATAIN
C      ALL NECESSARY DATA IS INPUT
C      CALL HJALG
C      THE PATTERN SEARCH ALGORITHM
C      CALL FINISH
C      CONCLUDE ALGORITHM IF MAXIMUM NUMBER
C      OF PATTERN SEARCH CYCLES REACHED
C
C      STOP
C      END

```



```

SUBROUTINE DATTAIN
  DIMENSION GAMMA(3,60),IDEL(3),SIR(3),SVLTT(3)
  DIMENSION EI(3),TII(3),CFG(60),CFI(3),W(3,6)
  DIMENSION TFG(50),TFI(3),WSUM(3,20)
  DIMENSION X(380),EPS(380),RHS(403),EPSY(7)
  DIMENSION CPT(3)
  DIMENSION EPSI(380)
  DIMENSION SIGN(590),IROW(590),MPRI(590),WEIGHT(590)
  COMMON/COMM01/NVAR,NOBJ,NPRIOR,NACH,NMAX
  COMMON/COMM02/X
  COMMON/COMM03/EPS
  COMMON/COMM04/RHS
  COMMON/COMM05/SIGN,WEIGHT,IROW,MPRI
  COMMON/COMM06/EPSY
  COMMON/COMM09/ALPHA,BETA
  COMMON/COMM10/MAXRED
  COMMON/COMM11/IPRINT
  COMMON/COMM12/GAMMA,IDEL,PCT,ITAU,XBETA,SVLTT
  COMMON/COMM13/SIR
  COMMON/COMM14/EPSI
  COMMON/COMM16/C,I,CT
  COMMON/COMM17/TFG,CFG
  INTEGER C,I,CT,CC
  INTEGER TC,CT2,CT3,CT4,CT5,T2
  READ(5,*) NVAR,NOBJ,NPRIOR,NACH,NMAX,IPRINT,MAXRED
  THE INITIAL GUESS FOR THE DECISION VARIABLES
  IS NOW READ IN. VARIABLES FROM THE MODEL ARE
  INPUT IN THE FOLLOWING ORDER: X'S, C'S, V'S, Y'S,
  U'S, W'S, E'S. ALL X'S ARE INPUT FIRST BY TIME
  PERIOD. FOR EACH TIME PERIOD VALUES FOR ALL
  SKILL TYPES ARE INPUT. AFTER ALL X'S ARE INPUT,
  ALL C'S ARE INPUT SIMILARLY. THIS IS FOLLOWED
  BY THE REMAINING DECISION VARIABLES.
  READ(5,*) (X(I),I=1,NVAR)
  EACH MODEL DECISION VARIABLE MAY BE WRITTEN IN
  IN TERMS OF THE GENERAL DECISION VARIABLE X,

```

C
C
C
C
C
C
C
C
C
C
C

```

C      THE NUMBER OF SKILL TYPES J, AND THE NUMBER OF
C      TIME PERIODS T.
C      MODEL VARIABLE
C      X
C      C
C      V
C      Y
C      U
C      W
C      E
C      WHERE I=1,2,...,C*T AND J=1,2,...,T
C      READ(5,*) (EPS(I),I=1,NVAR)
C      READ(5,*) ALPHA,BETA
C      READ(5,*) (EPSY(I),I=1,NPRIOR)
C      READ(5,*) ((SIGN(N),IROW(N),MPRI(N),WEIGHT(N)),N=1,NACH)
C      READ(5,*) C,T
C      CT=C*T
C      CT2=CT+CT
C      CT3=CT2+CT
C      CT4=CT3+CT
C      CT5=CT4+CT
C      T2=T+T
C      DO 70 I=1,T
C      70 READ(5,*) (GAMMA(CC,I),CC=1,C)
C      READ(5,*) (IDEL(CC),CC=1,C)
C      READ(5,*) (SIR(I),I=1,C)
C      READ(5,*) PCT
C      READ(5,*) ITAU,XBETA
C      READ(5,*) (SVLT(CC),CC=1,C)
C      READ(5,*) (CPT(CC),CC=1,C)
C      DO 71 I=1,NVAR
C      71 EPSI(I)=EPS(I)
C      EACH MODEL GOAL EQUATION SET MAY BE WRITTEN
C      IN TERMS OF THE GENERAL GOAL EQUATION Y, THE
C      NUMBER OF SKILL TYPES C, AND THE NUMBER OF
C      TIME PERIODS T. EQUATION SETS HAVE BEEN ORDERED

```



```

78 WSUM(CC,1)=W(CC,1)
   DO 80 CC=1,C
   L=IDEL(CC)
   DO 79 TC=2,L
79 WSUM(CC,TC)=WSUM(CC,TC-1)+W(CC,TC)
   L1=L+1
   DO 80 TC=L1,T
80 WSUM(CC,TC)=WSUM(CC,L)
   DO 86 TC=1,T
   DO 86 CC=1,C
86 RHS(CT+(TC-1)*C+CC)=CFG((TC-1)*C+CC)-CFI(CC)-
1 SVLTT(CC)*WSUM(CC,TC)
   RESERVE ACTIVATION TIME RIGHTHAND SIDE
   VALUES
   READ(5,*)((RHS(CT2+(TC-1)*C+CC),CC=1,C),TC=1,T)
   TECHNICAL TRAINING INSTRUCTORS RIGHTHAND
   SIDE VALUES
   READ(5,*)((TII(CC),CC=1,C)
   DO 181 CC=1,C
   RHS(CT3+CC)=SIR(CC)*TII(CC)
   DO 181 TC=2,T
181 RHS(CT3+(TC-1)*C+CC)=RHS(CT3+CC)
   INSTRUCTOR FLOWS TO CONUS RIGHTHAND SIDE
   VALUES
   DO 182 TC=1,T
   DO 182 CC=1,C
182 RHS(CT4+(TC-1)*C+CC)=PCT*TII(CC)
   INPUTS FOR TECHNICAL TRAINING RIGHTHAND
   SIDE VALUES
   KTAU=ITAU+1
   INPUT INITIAL BASIC TRAINING PIPELINE LEVELS.
   THIS IS ONE VALUE FOR EACH OF THE KTAU PREVIOUS
   CLASSES.
   READ(5,*)((EI(I),I=1,KTAU)
   DO 183 TC=1,KTAU
183 RHS(CT5+TC)=XBETA*EI(TC)

```

```

LTAU=KTAU+1
DO 184 TC=LTAU,T
184 RHS(CT5+TC)=0.
C      FLOW INTO BMT RIGHTHAND SIDE VALUES
C      READ(5,*) (RHS(CT5+T+TC),TC=1,T)
C      RESERVES TOTAL DRAWDOWN RIGHTHAND SIDE
C      VALUES
C      READ(5,*) (RHS(CT5+T2+CC),CC=1,C)
C      DESIRED CONJS PCT RIGHTHAND SIDE VALUES
C      DO 190 TC=1,T
C      DO 190 CC=1,C
C      I=(TC-1)*C+CC
190 RHS(CT5+T2+C+I)=RHS(CT+I)-(1.-3*F(CC))*3*F(I)
      RETURN
      END

```



```

C
C
C
SUBROUTINE VALUE(Z)
  SUBROUTINE VALUE EVALUATES THE ACHIEVEMENT
  FUNCTION GIVEN THE PREVIOUSLY EVALUATED
  GOAL EQUATIONS OF THE DECISION VARIABLES.
  DIMENSION Z(7), Y(403), RHS(403), SIGN(530), IROW(590)
  DIMENSION MPRI(590), WEIGHT(590)
  COMMON/COMM01/NVAR, NOBJ, NPRIOR, NACH, NMAX
  COMMON/COMM04/RHS
  COMMON/COMM05/SIGN, WEIGHT, IROW, MPRI
  COMMON/COMM07/Y
  CALL YVALUE
  DO 1 K=1, NPRIOR
    Z(K)=0.0
  1 CONTINUE
  DO 2 K=1, NACH
    DEV=Y(IROW(K))-RHS(IROW(K))
    IF(SIGN(K).LT.0) DEV=-DEV
    IF(DEV.LT.0) DEV=0.
    Z(MPRI(K))=Z(MPRI(K))+WEIGHT(K)*DEV
  2 CONTINUE
  RETURN
  END

```



```

C C KFG1 INDICATES WHICH PATTERN SEARCH PHASE
C C IS BEING PERFORMED
C C KFG1 VALUE
C C 0
C C SEARCH PHASE
C C FLOWS INTO TECHNICAL TRAINING
C C (V'S,W'S,U'S)
C C 1
C C SEARCH TO MEET MINIMUM PCT OF
C C CONUS GOALS (X'S)
C C 2
C C SEARCH TO REDUCE INITIAL CONUS
C C EXCESSES (Y'S)
C C
C C 100 CALL VALUE(Z)
C C
C C INITIAL K1 VALUE (IK1) IS THE SUBSCRIPT OF THE
C C FIRST V VARIABLE. LAST K1 VALUE (LK1) IS THE SUBSCRIPT
C C OF THE V VARIABLE OF THE NEXT TO LAST TIME PERIOD FOR
C C THE FIRST SKILL TYPE.
C C
C C IK1=CT+CT+1
C C LK1=IK1+CT-SHIFT(C,1)
C C LSUB2=LK1-SHIFT(C,1)
C C LSUB1=LSUB2-SHIFT(C,1)
C C 101 DO 150 K=1,NVAR
C C XBASE(K)=X(K)
C C 150 CONTINUE
C C
C C START THE SEARCH PATTERN OF FLOWS
C C INTO TECHNICAL TRAINING (V'S,W'S,U'S)
C C
C C DO 319 K1=IK1,LK1,C
C C K1=K1
C C K3=K1+C-1
C C DO 190 I=1,NVAR
C C 190 EPS(I)=EPSI(I)
C C NREDUC=0
C C KFLAG=1

```

```

KFG1=0
199 NIJK=0
C NIJK COUNTS THE NUMBER OF IMPROVEMENTS, IF ANY,
C DURING THE PATTERN SEARCH
C
200 DO 205 K=K1,K3
C
C THE ADJUSTMENT FIGURES (ADJ) WHICH FOLLOW ARE INTENDED
C FOR A THREE-SKILL, TWENTY-TIME-PERIOD MODEL. ADJ ASSOCIATES
C THE DESIRED PATTERN SEARCH STEP SIZE WITH A PARTICULAR
C SKILL TYPE. ADDITIONAL ADJUSTMENTS ARE REQUIRED FOR MODELS
C INVOLVING MORE SKILL TYPES.
C
ADJ=2.
IF(K.EQ.K1) ADJ=1.
IF(K.EQ.K3) ADJ=.2
IF(K.GE.LSUB1.AND.K.EQ.SHIFT((K1-K3,-1))) ADJ=0.
IF(K.GE.LSUB2.AND.K.EQ.K1) ADJ=0.
IF(K.GE.LK1.AND.K.EQ.K3) ADJ=0.
ADJ2=ADJ*EPS(K)
ADJ1=ADJ2/SIR(K-K1+1)
C STEP FORWARD AND EVALUATE
KFG=0
C
C KFG IS A SWITCH WHICH TELLS THE PROGRAM WHETHER THE
C PARTICULAR DECISION VARIABLE EQUALS ZERO OR NOT. IT IS
C USED TO PREVENT NEGATIVE VALUES FOR DECISION VARIABLES.
IF(X(K).EQ.0.) KFG=1
X(K)=X(K)+ADJ1
IF(K1.EQ.LK1) GO TO 201
X(K+CT2C)=X(K)
201 X(K+CT3C)=X(K+CT3C)+ADJ2
C SUBROUTINE DECIDE TAKES THE VALUE OF THE TEST POINT,
C EVALUATES THE ACHIEVEMENT FUNCTION VIA ZVALUE, AND
C DECIDES IF THE TEST POINT GIVES AN IMPROVEMENT.
C IJK=1 FOR IMPROVEMENT, IJK=0 IF NOT
CALL DECIDE(IJK)
NIJK=NIJK+IJK

```

```

C
IF(IJK.EQ.1) GO TO 205
STEP BACKWARDS AND EVALJATE
IF(X(K).EQ.ADJ1) X(K)=ADJ1+ADJ1
IF(K1.EQ.LK1) GO TO 202
X(K+CT2C)=X(K)
202 IF(X(K+CT3C).EQ.ADJ2) X(K+CT3C)=ADJ2+ADJ2
X(K)=X(K)-(ADJ1+ADJ1)
IF(K1.EQ.LK1) GO TO 203
X(K+CT2C)=X(K)
203 X(K+CT3C)=X(K+CT3C)-(ADJ2+ADJ2)
IF(X(K).EQ.0..AND.KFG.EQ.1) GO TO 205
CALL DECIDE(IJK)
NIJK=NIJK+IJK
IF(IJK.EQ.1) GO TO 205
X(K)=X(K)+ADJ1
IF(K1.EQ.LK1) GO TO 204
X(K+CT2C)=X(K)
204 X(K+CT3C)=X(K+CT3C)+ADJ2
205 CONTINUE
MCOUNT=MCOUNT+1
C CHECK IF MAXIMUM NUMBER OF SEARCH CYCLES EXHAUSTED
IF(MCOUNT.EQ.NMAX) GO TO 999
C CHECK TO SEE IF PATTERN RESULTED IN NO IMPROVEMENT
IF(NIJK.EQ.0..AND.KFLAG.EQ.1) GO TO 300
C ACCELERATION
DO 206 K=1,NVAR
IF(X(K).LT.0.) X(K)=0.
XSAVE(K)=X(K)
206 CONTINUE
DO 210 K=K1,K3
X(K)=X(K)+ALPHA*(X(K)-XBASE(K))
IF(K1.EQ.LK1) GO TO 207
X(K+CT2C)=X(K)
207 X(K+CT3C)=X(K+CT3C)+ALPHA*(X(K+CT3C)-XBASE(K+CT3C))
IF(X(K).LT.0.) X(K)=0.
IF(K1.EQ.LK1) GO TO 208

```



```

X(K+CT2C)=X(K)
208 IF(X(K+CT3C).LT.0.) X(K+CT3C)=0.
210 CONTINUE
DO 220 K=1,NVAR
XBASE(K)=XSAVE(K)
220 CONTINUE
C      NOW CHECK VALUE OF ACHIEVEMENT FUNCTION AT
C      ACCELERATION POINT
KFLAG=0
CALL DECIDE(IJK)
IF(IJK.EQ.0) KFLAG=1
IF(IPRINT.EQ.0) GO TO 299
WRITE(6,600) MCOUNT
WRITE(6,601)
DO 230 K=1,NPRIOR
WRITE(6,602) K,Z(K)
230 CONTINUE
WRITE(6,603)
IF(KFLAG.EQ.1) GO TO 270
DO 250 K=1,NVAR,4
WRITE(6,604) K,X(K),K+1,X(K+1),K+2,X(K+2),K+3,X(K+3)
250 CONTINUE
WRITE(6,605)
GO TO 199
270 DO 280 K=1,NVAR,4
WRITE(6,604) K,XBASE(K),K+1,XBASE(K+1),K+2,XBASE(K+2),K+3,XBASE(K+
1 3),K+4,XBASE(K+4)
C      PRINTS XBASE AS X-VALUE SINCE TEMPORARY SEARCH
C      POINT IS UNIMPROVED
280 CONTINUE
WRITE(6,607)
299 GO TO 199
C      REDUCE STEP SIZE
300 NPREDUC=NPREDUC+1
C      IF MAXIMUM NUMBER OF STEP SIZE REDUCTIONS EQUALED,
C      TERMINATE ALGORITHM

```

```

IF(NREDUC.GT.MAXRED) GO TO 315
IF(KFG1.NE.0) GO TO 304
DO 301 K=1,NVAR
X(K)=XBASE(K)
301 CONTINUE
304 DO 305 K=1,NVAR
EPS(K)=EPS(K)+BETA
305 CONTINUE
WRITE(6,605) NREDUC,MCOUNT
IF(IPRINT.EQ.0.AND.KFG1.EQ.1) GO TO 326
IF(IPRINT.EQ.0.AND.KFG1.EQ.2) GO TO 352
IF(IPRINT.EQ.0) GO TO 199
WRITE(6,601)
DO 314 K=1,NPRIOR
WRITE(6,602) K,Z(K)
314 CONTINUE
WRITE(6,603)
DO 315 K=1,NVAR,4
WRITE(6,604) K,X(K),K+1,K+2,X(K+2),K+3,X(K+3)
315 CONTINUE
IF(KFG1.EQ.1) GO TO 326
GO TO 199
316 WRITE(6,608)
IF(IPRINT.EQ.1) GO TO 599
WRITE(6,601)
DO 318 K=1,NPRIOR
WRITE(6,602) K,Z(K)
318 CONTINUE
IF(KFG1.EQ.1) GO TO 350
IF(KFG1.EQ.2) GO TO 360
WRITE(6,603)
DO 319 K=K1,K3
WRITE(6,604) K,XBASE(K),K+CT3,XBASE(K+CT3C),K+CT2C,XBASE(K+CT2C)
X(K)=XBASE(K)
X(K+CT3C)=XBASE(K+CT3C)
X(K+CT2C)=XBASE(K+CT2C)

```

```

319 CONTINUE
   KFG1=1
C
C      START THE SEARCH PATTERN TO MEET DESIRED
C      PERCENTAGE OF CONUS GOALS
C
      DO 325 I=1,NVAR
325 EPS(I)=2.
      NREDUC=0
326 NIJK=0
327 DO 340 K=1,CT
      IF(X(K).LT.EPS(K)) GO TO 340
      X(K)=X(K)-EPS(K)
      IF(K.GE.(CT-C+1)) GO TO 328
      X(K+C)=X(K+C)+EPS(K+C)
328 CALL DECIDE(IJK)
      NIJK=NIJK+IJK
      IF(IJK.EQ.1) GO TO 340
      X(K)=X(K)+EPS(K)
      IF(K.GE.(CT-C+1)) GO TO 340
      X(K+C)=X(K+C)-EPS(K+C)
340 CONTINUE
      MCOUNT=MCOUNT+1
      IF(MCOUNT.EQ.NMAX) GO TO 999
      IF(NIJK.EQ.0) GO TO 300
      GO TO 326

C
C      START THE SEARCH PATTERN TO REDUCE INITIAL
C      CONUS EXCESSES DUE TO RESERVES ACTIVATION
C
350 KFG1=2
      DO 351 I=1,NVAR
351 EPS(I)=EPSI(I)
      NREDUC=0
      CT3P1=SHIFT(CT,2)-CT+1
      CT3C8=SHIFT(CT,2)-CT+SHIFT(C,3)

```

```

352 NIJK=0
DO 355 K=CT3P1,CT3C8
IF(X(K).LT.EPS(K)) GO TO 355
X(K)=X(K)-EPS(K)
X(K+C)=X(K+C)+EPS(K+C)
CALL DECIDE(IJK)
NIJK=NIJK+IJK
IF(IJK.EQ.1) GO TO 355
X(K)=X(K)+EPS(K)
X(K+C)=X(K+C)-EPS(K+C)
355 CONTINUE
MOUNT=MCOUNT+1
IF(MOUNT.EQ.NMAX) GO TO 999
IF(NIJK.EQ.0) GO TO 300
GO TO 352
C      DETERMINE THE FLOWS INTO BASIC TRAINING
C      REQUIRED FOR THE FINAL SOLUTION.
360 ITAUS=T-ITAU-1
ISTEP=CT5+C*(ITAU+1)
DO 363 I=1,ITAUS
X(CT6+I)=0.
DO 362 J=1,C
362 X(CT6+I)=X(CT5+I)+X(ISTEP+J)
ISTEP=ISTEP+C
363 X(CT6+I)=X(CT5+I)/XBETA
ITAUS=ITAUS+1
DO 364 I=ITAUS,T
364 X(CT6+I)=0.
DO 365 I=1,T
365 Y(CT5+I)=X(CT6+I)
C
C      PREPARE AND PRINT SUMMARY OF RESULTS
C
DO 400 I=1,N0BJ
400 DEVN(I)=Y(I)-RHS(I)
C      A POSITIVE DEVIATION MEANS GOAL OVERACHIEVEMENT.

```

AD-A065 909

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 12/2
PERSONNEL CONTINGENCY PLANNING MODEL USING GOAL PROGRAMMING.(U)
DEC 78 J A MORENO, B W UTZ

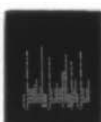
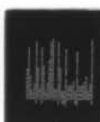
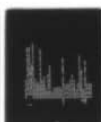
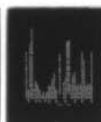
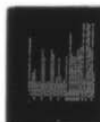
UNCLASSIFIED

AFIT/60R/SM/78D-10

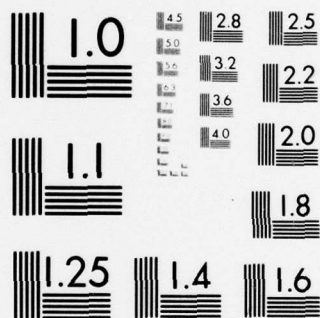
NL

4 OF 4

AD
A065909



END
DATE
FILMED
5-79
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

C      A NEGATIVE DEVIATION MEANS GOAL UNDERACHIEVEMENT.
      DO 425 I=1,CT
      TFL(I)=TFG(I)+DEVN(I)
425   CFL(I)=CFG(I)+DEVN(CT+I)
      WRITE(6,612)
      DO 430 TC=1,T
      DO 430 CC=1,C
430   WRITE(6,609) CC,TC,TFG((TC-1)*C+CC),DEVN((TC-1)*C+CC),CC,TC,
      1 TFL((TC-1)*C+CC),DEVN(CT+(TC-1)*C+CC)
      WRITE(6,613)
      DO 440 TC=1,T
      DO 440 CC=1,C
440   WRITE(6,610) CC,TC,CFG((TC-1)*C+CC),DEVN(CT+(TC-1)*C+CC),
      1 CFL((TC-1)*C+CC),DEVN(CT+(TC-1)*C+CC)
      WRITE(6,614)
      DO 450 K=1,NVAR,4
450   WRITE(6,604) K,X(K),K+1,X(K+1),K+2,
      1 X(K+2),K+3,X(K+3)
      WRITE(6,615)
      DO 460 K=1,NOBJ
460   WRITE(6,611) K,Y(K),K,RHS(K),DEVN(K)
599   STOP
600   FORMAT("-",25X,I3," SEARCH PATTERNS HAVE BEEN COMPLETED")
601   FORMAT("-",25X,"ACHIEVEMENT FUNCTION")
602   FORMAT("-",25X,"A(",I2,") = ",F13.6)
603   FORMAT("-",25X,"DECISION VARIABLES")
604   FORMAT("-",T10,4("X(",I3,") = ",F15.9,2X))
605   FORMAT("-",25X,"STEP REDUCTION NUMBER ",I3,
      1 " OCCURRED AFTER SEARCH PATTERN NUMBER ",I3)
606   FORMAT("-",25X,"KFLAG = 0 SO THAT THE TEMPORARY BASE POINT IMPROVE
      1S THE SOLUTION")
607   FORMAT("-",25X,"KFLAG = 1 SO THAT THE TEMPORARY BASE POINT DOES NO
      1T IMPROVE THE SOLUTION",/,",25X,"SHOULD THE PATTERN SEARCH FIND
      2NO IMPROVEMENT, STEP SIZES WILL BE REDUCED")
608   FORMAT("-",25X,"PATTERN SEARCH TERMINATED -- MAXIMUM NUMBER OF STE
      1P REDUCTIONS EQUALED")

```

```

612 FORMAT("-",25X,"THEATER FORCE LEVELS")
609 FORMAT(" ",T10,"TFG(",I3,"",I3,"") = ",F13.6,
1 3X,"TFL(",I3,"",I3,"") = ",F13.5,3X,"DEV = ",F13.6)
613 FORMAT("-",25X,"CONUS FORCE LEVELS")
610 FORMAT(" ",T10,"CFG(",I3,"",I3,"") = ",F13.6,
1 3X,"CFL(",I3,"",I3,"") = ",F13.5,3X,"DEV = ",F13.6)
614 FORMAT("-",25X,"DECISION VARIABLES")
615 FORMAT("-",25X,"Y(I),RHS(I),DEV(I)")
611 FORMAT(" ",T10,"Y(",I3,"") = ",F13.6,3X,
1 "RHS(",I3,"") = ",F13.6,3X,"DEV = ",F13.6)
999 RETURN
END

```

```

C      SUBROUTINE DECIDE(M)
C      SUBROUTINE DECIDE CALLS SUBROUTINE VALUE TO EVALUATE
C      THE ACHIEVEMENT FUNCTION AT THE TEST POINT AND
C      COMPARES THIS TO THE PREVIOUS BEST VALUE. IF THE
C      POINT WILL PROVIDE IMPROVEMENT, M=1 IS RETURNED
C      THRU THE ARGUMENT LIST. IF NOT, M=0 IS RETURNED.
C      DIMENSION X(380),Z(7),ZTEST(7),EPSY(7)
C      DIMENSION TFG(50),TFL(60),CFG(50),CFL(50)
C      DIMENSION DEVN(403),RHS(403),I(403)
C      COMMON/COMM01/NVAR,NOBJ,NPRIOR,NACH,NMAX
C      COMMON/COMM02/X
C      COMMON/COMM04/RHS
C      COMMON/COMM05/EPST
C      COMMON/COMM07/Y
C      COMMON/COMM08/Z
C      COMMON/COMM15/KK1
C      COMMON/COMM16/C,T,CT
C      COMMON/COMM17/TFG,CFG
C      COMMON/COMM18/DEVN
C      COMMON/COMM19/CFL,TFL
C      COMMON/COMM20/KFG1
C      INTEGER C,T,CT,TC,CC
C      CALL VALUE(ZTEST)
C      DO 10 K=1,NPRIOR
C      IF(ZTEST(K).GT.Z(K)+0.000001) 30 TO 90
C      IF(ZTEST(K)+0.000001.LT.Z(K)) 30 TO 20
C      10 CONTINUE
C      GO TO 90
C      20 DO 40 K=1,NPRIOR
C      TERMINATION BY MEANS OF ZERO FOR ALL PRIORITY
C      LEVELS
C      IF(ZTEST(K).NE.0.0) GO TO 50
C      40 CONTINUE
C      GO TO 100
C      50 DO 60 K=1,NPRIOR
C      TERMINATION TEST BY TOLERANCES

```

```

        IF (ABS(Z(K)-ZTEST(K)).GT.EPSY(<)) GO TO 70
60 CONTINUE
        IF (KK1.LT.(SHIFT(CT,2)+1-CT-SHIFT(C,1))) GO TO 70
        IF (KK1.GE.(SHIFT(CT,2)+1-CT-SHIFT(C,1)).AND.KFG1.NE.2) GO TO 70
        GO TO 100
70 DO 80 K=1,NPRIOR
        Z(K)=ZTEST(K)
80 CONTINUE
        M=1
        RETURN
90 M=0
        RETURN
100 WRITE(6,600)
    WRITE(6,601)
    DO 200 K=1,NPRIOR
        WRITE(6,602) K,ZTEST(K)
200 CONTINUE
C
C
C
        PREPARE AND PRINT SUMMARY OF RESULTS
C
        DO 250 I=1,NORJ
250 DEVN(I)=Y(I)-RHS(I)
        A POSITIVE DEVIATION MEANS GOAL OVERACHIEVEMENT.
        A NEGATIVE DEVIATION MEANS GOAL UNDERACHIEVEMENT.
        DO 300 I=1,CT
300 CFL(I)=CFG(I)+DEVN(I)
        TFL(I)=TFG(I)+DEVN(I)
        WRITE(6,605)
        DO 350 TC=1,T
        DO 350 CC=1,C
350 WRITE(6,606) CC,TC,TFG((TC-1)*C+CC),CC,TC,
            1 TFL((TC-1)*C+CC),DEVN((TC-1)*C+CC)
        WRITE(6,607)
        DO 400 TC=1,T
        DO 400 CC=1,C
400 WRITE(6,608) CC,TC,CFG((TC-1)*C+CC),CC,TC,

```



```

1 CFL((TC-1)*C+CC),DEVN(CT+(TC-1)*C+CC)
  WRITE(6,603)
  DO 450 K=1,NVAR,4
450  WRITE(6,604) K,X(K),K+1,X(K+1),K+2,X(K+2),K+3,X(K+3)
      WRITE(6,609)
  DO 500 K=1,NORJ
500  WRITE(6,610) K,Y(K),K,RWS(K),DEVN(K)
      STOP
600  FORMAT("-",25X,"SEARCH ALGORITHM IS COMPLETED -- TOLERANCES FOR IM-
      PROVED SOLUTIONS SATISFIED")
601  FORMAT("-",25X,"OPTIMAL ACHIEVEMENT FUNCTION VALUES")
602  FORMAT("-",25X,"A(",I2,") = ",F13.6)
603  FORMAT("-",25X,"OPTIMAL DECISION VARIABLE VALUES")
604  FORMAT("-",T10,4("X(",I3,") = ",F15.8,2X))
605  FORMAT("-",25X,"THEATER FORCE LEVELS")
606  FORMAT("-",T10,"TFG(",I3,") = ",F13.6,
1 3X,"TFL(",I3,") = ",F13.6,3X,"DEV = ",F13.6)
607  FORMAT("-",25X,"CONUS FORCE LEVELS")
608  FORMAT("-",T10,"CFG(",I3,") = ",F13.6,
1 3X,"CFL(",I3,") = ",F13.6,3X,"DEV = ",F13.6)
609  FORMAT("-",25X,"Y(I),RHS(I),DEV(I)")
610  FORMAT("-",T10,"Y(",I3,") = ",F13.6,3X,"RHS(",I3,
1  ") = ",F13.6,3X,"DEV = ",F13.6)
      END

```

```

C
C
C
SUBROUTINE YVALUE
  THE ORDERING OF GOAL EQUATION SETS HAS BEEN
  MODIFIED SLIGHTLY FROM THE THEORETICAL MODEL
  AS A PROGRAMMING CONVENIENCE.
  DIMENSION X(380),Y(403),GAMMA(3,50),IDEL(3),SIR(3)
  DIMENSION SVLTT(3)
  COMMON/COM402/X
  COMMON/COM407/Y
  COMMON/COM112/GAMMA,IDEL,PCT,IFAJ,XBETA,SVLTT
  COMMON/COM113/SIR
  COMMON/COM116/C,T,CT
  INTEGER T
  INTEGER TC,C,CC,CT,TC1
  INTEGER T2,CT2,CT3,CT4,CT5,CT5
  T2=T+T
  CT2=CT+CT
  CT3=CT2+CT
  CT4=CT3+CT
  CT5=CT4+CT
  CT6=CT5+CT
  C FIRST SET OF EQUATIONS--- OVERSEAS FORCE
  DO 100 TC=1,T
  DO 100 CC=1,C
  I=(TC-1)*C+CC
  Y(I)=X(I)-X(CT+I)
  IF(TC.EQ.1) GO TO 100
  GAMA=GAMMA(CC,TC)
  TC1=TC-1
  DO 90 IT=1,TC1
  Y(I)=Y(I)+GAMA*(X(I-C*IT)-X(I-3*IT+CT))
  90 GAMA=GAMMA(CC,TC-IT)*GAMA
  100 CONTINUE
  C SECOND SET OF EQUATIONS--- CONUS FORCE
  DO 200 TC=1,T
  DO 200 CC=1,C
  I=(TC-1)*C+CC

```

```

IF(TC.GT.IDEL(CC))GO TO 150
Y(CT+I)=X(CT+I)-X(I)-X(CT2+I)+X(CT3+I)+X(CT4+I)
IF(TC.EQ.1) GO TO 200
TC1=TC-1
DO 130 IT=1,TC1
130 Y(CT+I)=Y(CT+I)-X(I-C*IT)+X(CT+I-C*IT)-X(CT2+I-C*IT)
1 +X(CT3+I-C*IT)+X(CT4+I-C*IT)
GO TO 200
150 Y(CT+I)=X(CT+I)-X(I)-X(CT2+I)+X(CT3+I)+X(CT4+I)
1 +X(CT5+I-IDEL(CC)*C)*SVLT(CC)
TC1=TC-1
DO 154 IT=1,TC1
154 IF(IT.GT.(TC-IDEL(CC)-1)) GO TO 153
Y(CT+I)=Y(CT+I)-X(I-C*IT)+X(CT+I-C*IT)-X(CT2+I-C*IT)
1 +X(CT3+I-C*IT)+X(CT4+I-C*IT)+X(CT5+I-IDEL(CC)*C-C*IT)*
2 SVLT(CC)
GO TO 154
153 Y(CT+I)=Y(CT+I)-X(I-C*IT)+X(CT+I-C*IT)-X(CT2+I-C*IT)
1 +X(CT3+I-C*IT)+X(CT4+I-C*IT)
154 CONTINUE
200 CONTINUE
C THIRD SET OF EQUATIONS--- RESERVE ACTIVATION TIME
DO 250 TC=1,T
DO 250 CC=1,C
I=(TC-1)*C+CC
Y(CT2+I)=X(CT3+I)
250 CONTINUE
C FOURTH SET OF EQUATIONS--- TECH TN3 INSTRUCTORS
DO 350 TC=1,T
DO 350 CC=1,C
I=(TC-1)*C+CC
Y(CT3+I)=X(CT5+I)
IF(TC.EQ.1) GO TO 350
TC1=TC-1
DO 325 IT=1,TC1
325 Y(CT3+I)=Y(CT3+I)-SIR(CC)*(X(CT2+I-C*IT)-X(CT4+I-C*IT))

```

```

350 CONTINUE
C
C FIFTH SET OF EQUATIONS--- INSTRUCTOR FLOWS TO CONUS
DO 450 TC=1,T
DO 450 CC=1,C
I=(TC-1)*C+CC
Y(CT4+I)=X(CT4+I)
IF(TC.EQ.1) GO TO 449
TC1=TC-1
DO 425 IT=1,TC1
425 Y(CT4+I)=Y(CT4+I)+PCT*(X(CT4+I-3*IT)-X(CT2+I-C*IT))
449 CONTINUE
450 CONTINUE
C
C SIXTH SET OF EQUATIONS--- INPUTS FOR TECH FNG
DO 550 TC=1,T
CC=1
I=(TC-1)*C+CC
Y(CT5+TC)=X(CT5+I)
DO 500 CC=2,C
I=(TC-1)*C+CC
500 Y(CT5+TC)=Y(CT5+TC)+X(CT5+I)
IF(TC.LE.(ITAU+1)) GO TO 549
Y(CT5+TC)=Y(CT5+TC)-XBETA*(X(CT5+TC-ITAU-1))
549 CONTINUE
550 CONTINUE
C SEVENTH SET OF EQUATIONS--- FLOW INTO BMT
DO 575 TC=1,T
Y(CT5+T+TC)=X(CT5+TC)
575 CONTINUE
C
C EIGHTH SET OF EQUATIONS--- RESERVES TOTAL DRAWDOWN
DO 590 CC=1,C
TC=1
I=(TC-1)*C+CC
Y(CT5+T2+CC)=X(CT3+I)

```



```

DO 580 TC=2,T
I=(TC-1)*C+CC
580 Y(CT5+T2+CC)=Y(CT5+T2+CC)+X(CT3+I)
590 CONTINUE
C NINTH SET OF EQUATIONS--- DESIRED PCT OF CONUS GOALS
DO 595 TC=1,T
DO 595 CC=1,C
I=(TC-1)*C+CC
595 Y(CT5+T2+C+I)=Y(CT+I)
RETURN
END

```



```

SUBROUTINE FINISH
COMMON /COMM01/ NVAR,NOBJ,NPRIOR,NACH,NMAX
COMMON /COMM02/ X(380)
COMMON /COMM08/ Z(7)
WRITE(6,600)
WRITE(6,601)
DO 1 K=1,NPRIOR
WRITE(6,602) K,Z(K)
1 CONTINUE
WRITE(6,603)
DO 2 K=1,NVAR,4
WRITE(6,604) K,X(K),K+1,X(K+1),K+2,X(K+2),K+3,X(K+3)
2 CONTINUE
600 FORMAT("-",25X,"MAXIMUM NUMBER OF PATTERN SEARCH CYCLES EXCEEDED",
1/," ",25X,"BEST SOLUTION TO THIS POINT FOLLOWS")
601 FORMAT("-",25X,"ACHIEVEMENT FUNCTION VALUES")
602 FORMAT(" ",25X,"A(",I2,") = ",F13.5)
603 FORMAT("-",25X,"DECISION VARIABLES VALUES")
604 FORMAT(" ",I10,4("X(",I3,") = ",F15.8,2X))
RETURN
END

```

VITAE

Vitae

James Anthony Moreno was born on 12 April 1947 in Mount Clemens, Michigan. He graduated from high school in Norton, Ohio, in 1965 and attended the University of Akron from which he graduated in 1970 with a Bachelor's degree in Mechanical Engineering (magna cum laude) and a regular commission in the United States Army. He was initially assigned to the Second Armored Division, Fort Hood, Texas, as an infantry platoon leader. This was followed by assignments as operations officer and company commander in the 15th Ordnance Battalion, 60th Ordnance Group in Germany during the period 1972 through 1976. He completed the Ordnance Officer Advanced Course prior to entry into the Air Force Institute of Technology in 1977. He is married to the former Cheryl Lynn Schilling of Barberton, Ohio. They have two daughters, Kimberly and Andrea, and a son, James.

Permanent address: 676 E. Paige Avenue
Barberton, Ohio 44203

Bradley Wayne Utz was born 21 July 1951 in Hanover, Pennsylvania. He graduated from high school in Hanover, Pennsylvania, in 1969, and attended Lehigh University where he received the degree of Bachelor of Arts in Mathematics (summa cum laude) with interdepartmental honors in 1973.

While at Lehigh, he received a reserve commission in the United States Air Force in May 1973. On his initial active duty, he was assigned to the 20th Surveillance Squadron (ADCOM) to work as a space systems analyst and trainer. He received his regular commission in 1977 and entered the Air Force Institute of Technology in the same year. He is married to the former Shirley R. Bodily of Fort Walton Beach, Florida.

Permanent address: R.D. #6, Box 569
Hanover, Pennsylvania 17331

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/SM/GOR/78D-10	2. GOVT ACCESSION NO.	3. REPORT'S CATALOG NUMBER
4. TITLE (and Subtitle) PERSONNEL CONTINGENCY PLANNING MODEL USING GOAL PROGRAMMING		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
7. AUTHOR(s) James A. Moreno, Capt, USA Bradley W. Utz, Capt, USAF		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, OH 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE December 1978
		13. NUMBER OF PAGES 302
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Approved for public release; IAW AFR 190-17 JOSEPH P. HIPPS, Major, USAF Director of Information 19 Jan 79		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Operations Research	Computers	Personnel Management
Mathematical Programming	Computer Applications	Resources
Mathematical Models	Military Personnel	Military Requirements
Goal Programming	Allocations	Military Strategy
Algorithms	Planning	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
A military contingency planning model is designed, tested, and evaluated using goal programming analysis. This prototype is designed to aid the achievement of Continental United States (CONUS) and overseas theater requirements for personnel of different categories over an arbitrary length of time subject to resource and time constraints. The two alternative approaches of the research in goal programming are the linear		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

goal programming simplex method and the pattern search method. Tests are run on somewhat small examples--a three-job skill, six-time period case and six priority levels of projected goals and a three-job skill, twenty-time period contingency of seven priority levels of CONUS and overseas theater goals. The pattern search method (tailored to fit to the particular policy goal structure) appears to provide the analyst with reasonable results and computer resource conservation, whereas the simplex method provides for a mathematical global optimum solution at increased expense in computer resources.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)